

**PRACTICAL APPROACHES TO COMPLEX ROLE ASSIGNMENT
PROBLEMS**

IN ROLE-BASED COLLABORATION

by

LUMING FENG

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science (MSc) in Computational Sciences

The School of Graduate Studies

Laurentian University

Sudbury, Ontario, Canada

© LUMING FENG 2013

THESIS DEFENCE COMMITTEE/COMITÉ DE SOUTENANCE DE THÈSE

Laurentian University/Université Laurentienne
School of Graduate Studies/École des études supérieures

Title of Thesis Titre de la thèse	PRACTICAL APPROACHES TO COMPLEX ROLE ASSIGNMENT PROBLEMS IN ROLE- BASED COLLABORATION		
Name of Candidate Nom du candidat	Feng, Luming		
Degree Diplôme	Master of Science		
Department/Program Département/Programme	Computational Sciences	Date of Defence Date de la soutenance	August 22, 2013

APPROVED/APPROUVÉ

Thesis Examiners/Examineurs de thèse:

Dr. Ratvinder Grewal
(Co-Supervisor/Co-directeur de thèse)

Dr. Haibin Zhu
(Co-supervisor/Co-directeur de thèse)

Dr. Youssou Gningue
(Committee member/Membre du comité)

Dr. Lichuan Liu
(External Examiner/Examineur externe)

Approved for the School of Graduate Studies
Approuvé pour l'École des études supérieures
Dr. David Lesbarrères
M. David Lesbarrères
Director, School of Graduate Studies
Directeur, École des études supérieures

ACCESSIBILITY CLAUSE AND PERMISSION TO USE

I, **Luming Feng**, hereby grant to Laurentian University and/or its agents the non-exclusive license to archive and make accessible my thesis, dissertation, or project report in whole or in part in all forms of media, now or for the duration of my copyright ownership. I retain all other ownership rights to the copyright of the thesis, dissertation or project report. I also reserve the right to use in future works (such as articles or books) all or part of this thesis, dissertation, or project report. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that this copy is being made available in this form by the authority of the copyright owner solely for the purpose of private study and research and may not be copied or reproduced except as permitted by the copyright laws without written authority from the copyright owner.

ABSTRACT

Group role assignment (GRA) is an important task in Role-Based Collaboration (RBC). The complexity of group role assignment becomes very high as the constraints are introduced. According to recent studies, considerable efforts have been put towards research on complex group role assignment problems. Some of these problems are clearly defined and initial solutions are proposed. However some of these solutions were unable to guarantee an optimal result, or the time complexity is very high. In fact, many real world collaboration problems concern many types of constraints. Therefore, to make them practical, the accuracy and efficiency of the algorithms should be improved.

Role is the center of a role-based collaboration mechanism. Role plays a very essential part in the whole process of a collaboration system, without the roles, there would be no collaboration. One important function of the role is that it defines the features or requirements of a position which can be used to filter or access the candidates. The definition of roles greatly influences the evaluation results of candidates, which in turn influence the RBC algorithms significantly. Based on previous research, the role-based evaluation is associated with multiple attribute decision making (MADM). Role-based evaluation methods can be adopted from MADM methods. Selecting an appropriate method for a specific problem is difficult and domain oriented. Therefore, a dynamic evaluation model which can be expanded by domain experts and adapted to many cases is required. At present, there is limited research related to this requirement.

This thesis first focuses on two complex role-based collaboration problems. The first being group role assignment problems with constraints of conflicting agents, and the

second an agent training problem for a sustainable group. Practical solutions to these problems are proposed and resolved by IBM ILOG CPLEX. Simulations are conducted to demonstrate the performance of these solutions. From which I compare the solutions' performances with the initial solutions, and indicate the improvement of these proposed solutions. Secondly, this thesis clarifies the difficulties of connecting evaluation methods with real world requirements. In order to overcome these difficulties, I introduce an additional parameter, propose a dynamic evaluation model, and provide four synthesis methods to facilitate the requirements of a co-operation project which is funded by NSERC (Natural Sciences and Engineering Research Council of Canada).

The contributions of this thesis includes: clarifying the complexity of two complex role-based collaboration problem; proposing a better solution and verifying its efficiency and practicability; discussing the difficulties of connecting evaluation methods with real world problems; introducing an additional parameter to improve the accuracy of evaluation to some problems; proposing a role-based evaluation model to meet the requirements of adaptive and expandable.

Keywords—Role-Based Collaboration; Role Assignment; Optimization; Conflict resolution; Adaptive Assignment; Multiple Attribute Decision Making; Role-based Evaluation; Dynamic Evaluation Model.

ACKNOWLEDGEMENTS

Without the supervision and assistance of Dr. Haibin Zhu of Nipissing University, North Bay, this thesis would be very different. His willingness to improve the quality of role-based collaboration and his answers to my countless questions contributed greatly to the successes of my research. Robert Pickring and Doron Barak also deserve thanks for continuously helping on the evaluation modeling and testing.

I am extremely grateful to everyone on the advisory committee for their guidance. I would especially like to thank my supervisors, Dr. Haibin Zhu, Dr. Youssou Gningue, and Dr. R.S Grewal, for providing me with an overwhelming amount of support and investing considerable time and effort into this research, and providing me with many excellent suggestions. Without their assistance, this thesis would not have been possible.

Furthermore, I would like to thank IBM Academic Initiative Program for facilitating the IBM ILOG CPLEX Optimization Studio without any charge, and Collaborative Lab, Nipissing University, NSERC for providing funding and computer hardware on which I was able to run the tests of my solutions.

Finally, I am grateful for all of my family members for great support throughout the period of my graduate studies at Laurentian University and Nipissing University in Canada.

TABLE OF CONTENTS

Abstract	iii
Acknowledgements	v
Table of Contents	vi
List of tables	x
List of figures	xii
1. Introduction	1
1.1 Role-based collaboration	1
1.2 Role-based collaboration problem types and examples	2
1.2.1 Basic RBC problem	3
1.2.2 Multi-agents RBC problem	3
1.2.3 Group role assignment problem with constraints of conflicting agents	4
1.2.4 Agent training for a sustainable group problem	5
1.3 E-CARGO model for role-based collaboration	6
1.4 Evaluation in RBC	7
2. Literature review	10
2.1 Related works of role-based collaboration and complex RBC problems	10
2.1.1 Related works of group role assignment with conflict agents	11
2.1.2 Related works of agents training problem for a sustainable group	13
2.2 Related works of role-based agent evaluation	15

2.2.1	Evaluation in role-based collaboration	16
2.2.2	Common MADM methods	18
2.3	Summary	22
3.	Problem statements and initial solutions	23
3.1	Related concepts	23
3.1.1	What is a role?	23
3.1.2	Role-based collaboration	26
3.1.3	The E-CARGO model	29
3.2	Problem definitions	38
3.2.1	Basic RBC problem	38
3.2.2	Multi-agents RBC problem	40
3.2.3	Group role assignment problem with constraints of conflicting agents	41
3.2.4	Agents training problem for a sustainable group	45
3.3	Initial solutions	53
3.3.1	Initial solution for basic RBC problem	53
3.3.2	An initial solution for multi-agents RBC problem	55
3.3.3	An initial solution for the group role assignment problem with constraints of conflicting agents	58
3.3.4	An initial solution of the Agents training problem for a sustainable group	59
3.4	Summary	60

4.	Solving the complex role-based collaboration with linear programming	61
4.1	Express and solve the RBC problem as linear programming problem	61
4.1.1	Transfer RBC problem to linear programming	61
4.1.2	Tool for resolving linear programming	63
4.2	Solutions for group role assignment problem with constraints of conflicting agents	64
4.2.1	Transfer the group role assignment problem with constraints of conflicting agents (role level) to LP and resolve with IBM ILOG.	64
4.2.2	Transfer the group role assignment problem with constraints of conflicting agents (group level) to LP and resolve with IBM ILOG.	80
4.3	Solution for agents training problem for a sustainable group	82
4.3.1	Transfer the agents training problem to linear programming problem and resolve with IBM ILOG	82
5.	Evaluation in practical RBC solution for project management	92
5.1	Introduction of the project	92
5.2	Evaluation model design for the CMCI project	93
5.3	Synthesis methods:	98
5.3.1	Method 1: Simple Additive Weighting (SAW)	99
5.3.2	Method 2: Multiplicative Exponent Weighing (MEW)	99
5.3.3	Method 3: Weighted Distance (WD)	100
5.3.4	Method 4: Weighted Area (WA)	101

5.4 Simulation	102
6. Conclusion	104
7. Future work	107
References	110
Published or accepted papers	115
Appendices	116
Appendix I: An AHP example	116
Appendix II: Hungarian(K-M) Algorithm solution in JAVA	121
Appendix III: Solution of GRACCA(role level) and simulation program	127
Appendix IV: Solution of GRACCA (group level)	133
Appendix V Solution of Agents training problem for a sustainable group and simulation program	137
Appendix VI: Solution for role-based evaluation and simulation program	142
Appendix VII: Simulation of agents information	150
Appendix VIII: Simulation result (SAW method)	151
Appendix IX: Simulation result (MEW method)	152
Appendix X: Simulation result (weighted distance method)	153
Appendix XI: Simulation result (weighted area method)	154

LIST OF TABLES

Table 3.1 Cost matrix of the restaurant

Table 3.2 The players' training cost indices

Table 3.3 The players' training cost indices

Table 3.4 Original qualification matrix and role range vector

Table 3.5 Transferred qualification matrix and matching array

Table 3.6 Transferred assignment matrix and matching array

Table 3.7 Compressed assignment matrix and role range vector

Table 4.1 Configurations of test environment

Table 4.2 Test results of ILOG solution and Hungarian solution

Table 5.1 Data structure design of role requirement

Table 5.2 The simulation of role requirements

Table A.1 Candidates information

Table A.2 AHP scale weights

Table A.3 Criteria weights for choosing a PM

Table A.4 Priorities of criterions for choosing a PM

Table A.5 Pairwise compare of experiences and calculated priorities

Table A.6 Pairwise compare of Educations and calculated priorities

Table A.7 Pairwise compare of Charismas and calculated priorities

Table A.8 Pairwise compare of Ages and calculated priorities

Table A.9 Calculation of the criterions priorities for each candidate

Table A.10 Calculation of the general priorities for each candidate

Table A.11 The simulated agents information

Table A.12 Evaluation results calculated by the simple additive weighting method

Table A.13 Evaluation result calculated by multiplicative exponent weighting method

Table A.14 Evaluation result calculated by weighted distance method

TableA.15 evaluation result calculated by weighted area method

LIST OF FIGURES

Figure 3.1 Role as a wrapper of a person in collaboration

Figure 3.2 A group and its role assignment matrix.

Figure 3.3 An ability matrix and an assignment matrix T .

Figure 3.4 A role range matrix Π and an ability matrix for a sustainable group with II .

Figure 3.5 A Ω Matrix and two Q Matrices.

Figure 4.1 Transfer the requirement constraints to ILOG expressions.

Figure 4.2 Transfer the unique task constraints to ILOG expressions.

Figure 4.3 Transfer the conflict constraints to ILOG expressions.

Figure 4.4 Assignment matrix T

Figure 4.5 Assignment result from ILOG solution

Figure 4.6 Time cost table of ILOG solution for group role assignment problem with constraints of role level conflicting agents (A)

Figure 4.7 Time cost table of ILOG solution for group role assignment problem with constraints of role level conflicting agents (B)

Figure 4.8 Time cost table of ILOG solution for group role assignment problem with constraints of role level conflicting agents (C)

Figure 4.9 Time cost table of ILOG solution for group role assignment problem with constraints of role level conflicting agents (D)

Figure 4.10 Transfer the group level conflict to linear programming constraint

Figure 4.11 Declare and add the objective in ILOG

Figure 4.12 Adding constraints in ILOG.

Figure 4.13 Resolve the adaptive collaboration in ILOG.

Figure 4.14 Compressing the result matrix from ILOG to final assignment matrix T.

Figure 4.15 Trends of time cost under regular configuration.

Figure 4.16 Trends of ratio simulation 1, $n/m=1/3$.

Figure 4.17 Trends of ratio simulation 1, $n/m=1/2$.

Figure 4.18 Sustainable simulation 1, $m=100, n=50$.

Figure 4.19 Sustainable simulation 2, $m=100, n=25$.

Figure 5.1 Evaluation model

Figure 5.2 UML of CMCI project

Figure A.1 AHP hierarchy for choosing PM

1. INTRODUCTION

1.1 Role-based collaboration

Nowadays, a society is organized in a hierarchical structure. For example, a country consists of provinces which themselves consist of districts. Similarly, a company is composed of departments and each department is composed of staff. The purpose of a hierarchical structure is to increase the efficiency of management and productivity of an organization in a general sense. In other words, an organization is collaborating with its sub-level entities. This is especially required and essential when one sub-level entity cannot complete a task alone. Collaboration (or group work) requires that participants fulfill their obligations and respect other participants' rights. It requires that everybody respects the social laws of the community [1]. To collaborate, people generally join a group or organization. All individuals should have clear positions within a group and their roles should be related but should not interfere with each other. Therefore, individuals fill specific positions and play specific roles in an organization. The roles played by the participants in a collaborative activity are important factors in achieving successful outcomes, and the role concept is the primary concern in the development of computer-based collaboration systems. The concept of roles will be discussed in Chapter 2. Collaboration is a common activity in a society of participants. However, there are many conflicts or constraints among the participants and roles which bring about lots of side effects to the organization's productivity. The question of how to achieve the most productive collaborations by manipulating role assignments and the configuration of teams is still largely unexplored.

Role-based collaboration is a recent innovation that addresses these concerns. It is a new methodology developed to organize collaboration by providing mechanisms for role specification, assignment, transition, and negotiation. With these mechanisms, individuals can have a clear understanding of the roles in the collaboration, thereby making the collaboration more productive. Based on the discoveries and demands of managerial, social, and psychological science [1], [2], [3], [4], a role-based collaborative system allows users to improve the productivity of collaboration by negotiating, tuning, and transferring relevant roles in a system (the properties of role-based collaboration are discussed in Chapter 2). Even though the concepts and aims of role-based collaboration (hereafter RBC) are clear, there are still numerous challenges due to the variety of the problems faced when implementing such a system in order to solve real-world collaboration problems. The most significant one is the efficiency of the assignment algorithm. In some complex collaboration issues, the solutions provided by early researchers are theoretical but not applicable. Therefore, how to clarify these problems and resolve them is important to the RBC.

1.2 Role-based collaboration problem types and examples

Role-based collaboration is highly connected with real world problems, there are many types of RBC problems that can be defined. The difficulty level of an RBC problem can be analyzed from different perspectives. From the scale perspective, it can be as simple as two positions needing to be filled by two people, or it can be as complicated as thousands of positions needing to be assigned within thousands of candidates. Thereby, the size of an RBC problem should be considered. From the scenario perspective, RBC

problems can be categorized by their complexities. This thesis discusses and provides solutions to the following RBC problems.

1.2.1 Basic RBC problem

The basic RBC problem is used to describe the scenario that there are n candidates and m roles that need to be assigned, and each role requires only one agent (m and n are integers, and $n \geq m$). This type of problem is the simplest scenario among all RBC problems and is the base of other types of RBC problems in this thesis. It can be applied in many problems, scheduling is one such problem. For example, in a hospital, the doctors' time slots need to be filled by different patients based on their preferences.

1.2.2 Multi-agents RBC problem

The Multi-agents RBC problem is almost similar to the basic one. It describes a situation that in an environment, there are n candidates and m roles that need to be filled. Each role may require more than one agent to take. The sum of required agents for all roles should be smaller than the number of candidates. If this is not the case, an insufficient condition will occur. This type of RBC problem is frequent in many industries, because on many occasions, one person is unable to assume all responsibilities, or unable to complete all the tasks of a role. Put simply, a specific role may request more than one agent (person) to make it functionally working. For example, a football team

needs 2 forwards, 4 middle fielders, 4 back fielders and 1 goal keeper, the roles of “forward”, “middle field” and “back field” need multiple agents.

1.2.3 Group role assignment problem with constraints of conflicting agents

The Group role assignment problem with constraints of conflicting agents (GRACCA) is a specific Multi-agents RBC problem that considers the conflicts amongst candidates. In the real world, conflicts are everywhere. For example, in the storage industry, if two types of chemicals react, we cannot place them into the same container. The purpose of RBC is to maximize (or minimize) the outcome (or cost) of the group, but the conflicts may have a negative effect or even a great influence on achieving the goal. Hence, how to overcome the negative effect of the conflicts and ensure the best outcome is the key point of this type of RBC problem. Conflicts can be considered from different perspectives, they could appear either in the role level or in the group level. For the role level conflict, any two candidates having conflict cannot be assigned on the same role. The goal here is to remove the conflicts within every role. The group level conflict means that any two of the candidates in the group cannot have any conflict. The goal for this is removing conflicts in the organization. In general, wherever the conflicts are considered, they make the collaboration much more complicated. However, this type of RBC problem is found, and needs to be solved, in many managerial tasks, e.g., human resource management or project management.

1.2.4 Agent training for a sustainable group problem

Considering that a group is defined by n roles and m agents, every agent has a training cost on each role. The group needs to have the potential ability of adapting several combinations where numbers of requested agents on roles are various. The question is, how goes one find an assignment that sustains all the combinations and ensures that the cost for the selected agents are the most economic amongst all possible assignments? This scenario is referred to as the Agents training problem for a sustainable group. It aims to find the best assignment adapting a certain number of specified combinations. This type of RBC problem is normally required in some agile situations, like a football manager wanting to select players that are adaptive to several formations.

The first two types of RBC problem can be solved in a polynomial time by applying the Hungarian algorithm. The corresponding solutions will be demonstrated in Chapter 3. Dr. Zhu provides initial solutions for the last two types of complex RBC problems based on the Hungarian algorithm, but the time complexity of the initial solution for a group role assignment problem with constraints of conflicting agents is extremely high, and the solution of the agents training problem for a sustainable group can only obtain a compromising result. Thereby, both of the initial solutions are hard to apply to real world problems. Therefore, solving these two types of complex RBC problems is a primary goal of my research. The definitions and initial solutions of these two problems will be shown in Chapter 3. My proposed solutions for them are demonstrated in Chapter 4.

1.3 E-CARGO model for role-based collaboration

Before we start exploring the RBC related problems, we need a formal model to clearly describe the problems. The E-CARGO model is introduced to support RBC. It means Environments, Classes, Agents, Roles, Groups and Objects [5]. The E-CARGO model is designed and established by Dr. Haibin Zhu, which formalizes the components of a role-based collaborative system and defines the internal structures.

The E-CARGO model clearly specifies the responsibilities and relationships of each component in a role-based collaborative system, provides the robustness, efficiency and correctness, and guarantees a successful design and implementation of the entire system [5]. The following advantages are also included in the E-CARGO model:

- A role is independent of groups and it can be taken as the central mechanism to support collaboration.
- A role is a manageable entity that can be flexibly modified and tuned.
- Users have the flexibility to have agents help their collaborative work or accomplish the jobs on their behalf.
- A group is specified based on roles. This facilitates the group management tasks.

My research is generally based on the E-CARGO model. I extend it in cases to define the complex RBC problems or other aspects of the RBC, e.g., evaluation. The details of the E-CARGO model will be presented in Chapter 2.

1.4 Evaluation in RBC

During my research, I was given the opportunity to participate in an NSERC funded project that primarily focused on developing an optimization method to find the best assignment in a project management system. I believe that my solutions for the RBC related problems can provide a solution to this problem. During the implementation of my algorithms, I found the agent evaluation to be an important and key issue for collaboration. Because collaboration is designed to obtain optimal group performance, to achieve this goal, we need to evaluate individual performance of each member on every role in a group. A better evaluation result (more accurate or appropriate), therefore leads to a better optimal result. But in a general perspective, the difficulties of evaluation include:

- 1) It is complex and hard to set up a well-accepted method or approach to evaluate agents.
- 2) The evaluation is domain-oriented and task oriented. One method that is useful in a specific domain or task may not be appropriate in other domains or tasks [6].

In the context of role-based collaboration, even though agent evaluation is still challenging and difficult, there is twilight that gives us a hope to provide a more appropriate model and method to evaluate agents. The starting point is to use roles as the standard to evaluate agents. We argue that it is inappropriate to evaluate an agent in a general context but for a role. We can say if an agent is good or bad in playing one role, but we cannot say that an agent is good or bad in general. For example, a person, Kobe Bryant, is highly qualified to take the role as the basketball player. However, he is not qualified to take a role as a programmer in a software development team. In other words,

the evaluation of an agent in role-based collaboration is role sensitive. Therefore, based on the E-CARGO model, I propose a new structure to support agent evaluation and role-based collaboration. This method requires a clear role requirement definition and a detailed agent ability evaluation. It can be applied in cases where the role requirement can be split into atomic ability requirements. And each ability can be valued or asserted. In Chapter 2, multiple criteria decision making methods related to evaluation are discussed, and in Chapter 5, my model of evaluation for role-based collaboration is illustrated.

In summary, this thesis presents my research in the field of role-based collaboration. I mainly focus on two aspects of the RBC process: 1) I investigate two kinds of complex optimization problems, provide practical solutions and analyze the results by comparing with previous researchers' works. 2) I discuss the evaluation in RBC, propose an evaluation model adapting to a real world problems, and make the solutions for complex RBC problems applicable.

This thesis is arranged as follows:

Chapter 2 reviews the related previous researchers' achievements and related methodologies; Chapter 3 introduces the concepts of role-based collaboration and a model called "E-CARGO" which is modeling the RBC. This chapter also clarifies the problem statements for the four types of RBC problems mentioned in this chapter and demonstrates the initial solutions from previous researchers. Chapter 4 presents my approaches to solving complex RBC problems, including a group role assignment problem with constraints of conflicting agents and an agents training problem for a sustainable group; Chapter 5 first introduces a practical project and discusses the

requirements for a dynamic evaluation model. It then demonstrates a dynamic role-based evaluation model proposed by myself. Chapter 6 concludes my research and proposes related and unexplored issues that need to be solved in future.

2. LITERATURE REVIEW

Role-Based Collaboration (RBC) is an evolving methodology designed to facilitate an organizational structure, provide logical system behavior, and enhance system security for both human and non-human entities that collaborate and coordinate their activities with or within systems [9]. The life-cycle of RBC includes three major tasks: role negotiation, assignment and performance [5]. Role assignment is, obviously enough, an important aspect of RBC. It greatly influences the efficiency of collaboration and the satisfactory degree among members of organizations that are involved in the collaboration. To better understand it, role assignment process can be split into three parts: agent evaluation, group role assignment, and role transfer [10]. As mentioned in the previous chapter, my research concentrates on the agent evaluation and the algorithms for complex group role assignment problems. This chapter demonstrates the related works of previous researchers and some methodologies that are associated with RBC.

2.1 Related works of role-based collaboration and complex RBC problems

Collaboration is working with each other to accomplish a task. It is a recursive process where two or more people or organizations work together to realize shared goals. In the field of role-based collaboration, due to the variety of the scenarios, there are many kinds of complex collaboration problems that can be defined with the concepts of RBC. I investigate two complex RBC problems which are largely based on Dr. Haibin Zhu's previous research. For these two complex RBC problems, little research

is conducted by other researchers. However, some researchers contribute from different perspectives.

2.1.1 Related works of group role assignment with conflict agents

Conflict resolution is one of the most important problems that must be solved when building a collaborative system. However, there is little research on conflict management using role-based methods.

Bhardwaj and Chandrakasan present a real world application that requires role assignment [11]. They introduce a role assignment framework and use it to derive bounds of the lifetime of a sensor network for a variety of data gathering scenarios that can be used to promote general role assignment algorithms.

Chaimowicz et al. [16] suggest a methodology for coordinating multiple robot teams in the execution of cooperative tasks. It is based on a dynamic role assignment mechanism in which the robots assume and exchange roles during the process of cooperation. They demonstrate the role assignment under a hybrid system framework, using a hybrid automaton to represent roles, transitions and controllers. A robot team is a good application of the proposed algorithm for dynamic assignment.

In Agent-Oriented Software Engineering, Ferber et al. [3] propose that multiple agent systems should be designed in an organization-centered way. They accentuate the importance of a group structure with role definitions in designing a multiple agent system. Role assignment turns out to be an important job in such a design methodology.

Nyanchama and Osborn [16] designate a role-graph model for role-based access control (RBAC). They use the graph model to provide taxonomy for different kinds of conflicts in role assignment. They simplify the complex problem of role assignment in consideration of role conflicts by partitioning the role graph into non-conflicting collections that can together be assigned to an agent.

Odell et al. [13] point out that the roles played by an agent may change over time. They conduct a case study where such role changes are required, analyzing and classifying the various types of role changes that may occur. Their contribution focuses on the third step of role assignment, i.e., role transfer.

Stone and Veloso [14] introduce periodic team synchronization (PTS) domains as time-critical environments in which agents act autonomously. They point out that dynamic role adjustment makes possible formation changes allowing for a group of agents to collaborate. They apply their method to a robot soccer team and gain a convincing result.

Vail and Veloso [9], [15] extend in role assignment and coordination in multi-robots system, especially in highly dynamic tasks. They develop an approach to sharing sensed information and effective coordination through the introduction of shared potential fields. The potential fields were based on the current positions of the other robots on the team and the soccer (in a robot soccer team).

Haibin Zhu is the first to clarify the specifications of role assignment problems with possible conflicting agents in the group context. He also proposes an exhaustive search method and an improved algorithm based on the exhaustive search method.

Simulations and experiments are also conducted to reveal that the time complexities of the proposed methods are very high, around $O(2^n)$ [18]. Thereby, these methods are not practical.

Therefore, an efficient and practical algorithm for solving the group role assignment(GRA) problem with constraints of conflicting agents is necessary.

2.1.2 Related works of agents training problem for a sustainable group

Like the example from the introduction chapter, sustainability is very important in our life, and a group structure could indeed assist in the establishment of sustainable collaboration. However, limited comprehensive research is conducted in the area of collaborative technologies and systems. A few related contributions are found in different fields.

Dyllick, and Hockerts [19] illustrate three key points for corporate sustainability: integrating the economic, ecological and social aspects in a ‘triple-bottom line’; integrating the short-term and long-term aspects; and consuming the income and not the capital. The agent training plan in this thesis follows the second key point.

Kumar and van Dissel [20] investigate the influence of Information Technology (IT) on collaboration. They identify the possible risks of conflict in the inter-organizational systems. They indicate that the IT enabled cooperation can degenerate into conflict. This point shows the importance of the agents training problem for a sustainable

group. The training plan contributes to conflict avoidance in collaboration among agents in a dynamic environment.

Rulke and Galaskiewicz [21] investigated the effect of knowledge distribution and group structure on performance in Master of Business Administration (MBA) game teams. They found that the group performance was depending on the distribution of knowledge within the group and networks of social relationships among group members. Their findings demonstrate that agent evaluation and role assignment are the key issues for group performance.

Sandholma [10] points out that in order to achieve outstanding collaboration results, it is necessary to tackle quality in a structured and long-term way and concentrate on thorough planning. A strategic plan containing all quality related activities has to be developed by combining different concepts, approaches, and methods in such a way that they will lead to good competitiveness and excellent collaboration results.

Faustmann [24] proposes an approach that configures parts of a detailed process model with different support strategies, i.e., task distributions. The configuration model allows workflows to be organized in a hierarchy structure to facilitate adaptation.

In the research of Adaptive Workflow Systems (AWfSs), Kammer et al. reviewed specific approaches to support the design of an adaptive workflow infrastructure [23]. The general goals of AWfSs are detecting, avoiding, and handling exceptions. The main functionality of an AWfS is to dynamically adjust workflow to adapt to changing job requirements. Muller et al. proposed AGENTWORK for a predictive adaptation which can preemptively adapt the unexecuted parts of running workflows in a largely automated

manner. The above contributions to AWfSSs present solid groundwork for the establishment of role relationships required in the proposed research [25].

In adaptive computing and services, Berman et al. showed that schedule adaptation assisted the GRID environment to provide available services when resource availabilities change [26]. Makris et al. provided an efficient way to compose Web services into new ones [27]. Finally, both Berman et al. and Makris et al. provided basic methods to deal with the constraints in role assignment.

In adaptive collaboration, H. Zhu, M. Hou, and M.C. Zhou clarified the parameters for group performance and established a solid mathematical foundation for adaptive collaboration. They also proposed an initial heuristic solution to the adaptive collaboration. However, the optimality is not satisfied. In some cases, the optimality is only 64%, which is not acceptable for many applications. Therefore, an accurate and practical solution for this problem is required.

2.2 Related works of role-based agent evaluation

Role-based agent evaluation is used to esteem the qualification of an agent for a role. It is required to check the capabilities, experiences, and credits of agents based on role specifications. However agent evaluation is rarely found in the literature of agent field [30].

Moore et al. [31] mentioned such problems but demonstrated ideas in a different way. They discussed a list of problems in selecting agents to execute a specific task. They

presented initial thoughts by considering roles and capabilities, but failed to fully clarify the problem of agent evaluation.

2.2.1 Evaluation in role-based collaboration

Role-based collaboration aims to provide the assignment that provides the best total output from all possible assignments. Normally, RBC related algorithms work with the qualification matrix as the input and calculate the assignment matrix based on it. Because of this, providing an accurate qualification matrix is vital to the RBC process.

According to the definition of RBC, each element of the qualification matrix is role based, which indicates the capability and compatibility of a specified agent to a specified role. Therefore, the evaluation of agents is role based too. When compared with normal general evaluation methods, it has the following advantages:

- It is rational and much more accurate in circumstances that the features of roles are various.
- The concept of role based evaluation is clear and understandable. From the qualification matrix, the values of different agents on the same role and one agent's evaluations on different roles are comparable.

Also, there are some disadvantages:

- Role-based evaluation requires significantly more computation time. Because each agent has to be evaluated on every role. Instead, the general evaluation

just gives one evaluation value of an agent regardless of how many roles are there in the circumstance.

- The process of defining roles is complicated and case sensitive. For different roles, the number of criteria and the way to define the criteria itself are various.

Based on the definitions of RBC, methods fulfilling the requirement of the role-based evaluation are required. By studying the nature of role-based evaluation, I find the key point is role definition, which consists of numerous requirements (criteria). The concept of role-based evaluation is connected with a knowledge domain of multiple criteria decision making (MCDM) which makes decisions in the presence of multiple, conflicting, and independent criteria. MCDM problems are commonly categorized by continuous or discrete based on the domain of alternatives with the problem. Hwang and Yoon classify them as:

- a) Multiple Attribute Decision Making (MADM), with discrete, usually limited, number of pre-specified alternatives, requiring inter and intra-attribute comparisons, involving implicit or explicit tradeoffs [32];
- b) Multiple Objective Decision Making (MODM), with decision variable values to be determined in a continuous or integer domain, of infinite or large number of choices, to best satisfy the decision making (DM) constraints, preferences or priorities [32].

In my research, I focus on MADM, as the alternatives of MADM are discrete and normally pre-defined; it is similar to the problem I am approaching with RBC.

Churchman is one of the early academics resolving the MADM problem formally by using a simple additive weighting method. Over the years, many methodologies are proposed by behavioral scientists, mathematicians, operational researchers, and decision theorists. Gershon and Duckstein state that the major criticism of MADM methods is that different techniques yield different results when applied to the same problem, apparently under the same assumptions and by a single DM [33]. Voogd found that in at least 40% of cases, each technique produced a different result from a separate technique [34]. The inconsistency of the various results is caused by:

- a) The techniques use weights differently in their calculations;
- b) Algorithms differ in their approach to selecting the ‘best’ solution;
- c) Many algorithms attempt to scale the objectives, which affects the weights already chosen;
- d) Some algorithms introduce additional parameters that affect the chosen solution.

Some MADM methods may appear to be suitable for a particular problem and may appear very different for another problem; the variety of available methods confuses the end user a lot. And the user faces the task of selecting the most appropriate method from among all feasible methods. I will introduce several common MADM methods throughout this chapter.

2.2.2 Common MADM methods

Simple Additive Weighting (SAW)

Simple additive weighting is the best known and most widely used method which transforms a vector to an appropriate scalar value based on the effectiveness of each attribute. Assume that there are n attributes, the SAW method uses all the n attribute values of alternatives and uses the regular arithmetical operations of multiplication and addition, and the attribute values must be both numerical and comparable. The mathematic format of SAW is:

$$S_i = \sum_j c_j \times r_{ij}$$

where, i is the index of agent, j is the index of role requirement, S_i is the evaluation result of agent i , c_j is the weight of the role requirement j , r_{ij} is the ability value of agent i on role requirement j .

Multiplicative Exponent Weighting (MEW)

MEW is a theoretically attractive contrast against SAW. Due to its mathematic concept, it does not attract the majority of practitioners and has not been applied in many circumstances. The mathematic format of MEW is:

$$S_i = \prod_j r_{ij}^{c_j}$$

where, i is the index of agent, j is the index of role requirement, S_i is the evaluation result of agent i , c_j is the weight of the role requirement j , r_{ij} is the ability value of agent i on role requirement j .

Technique for Preference by Similarity to the Ideal Solution (TOPSIS)

TOPSIS is a technique for Order Preference by Similarity to the Ideal Solution [32]. It defines an index called similarity to the positive ideal solution (proximity to positive and remoteness to negative values). Normally the ideal solution is unattainable or infeasible. When an alternative moves away from this ideal, the DM's utilities decrease monotonically. The TOPSIS method is normally defined in 6 steps [40], [41], [42]:

1. Calculate the normalized ratings.
2. Calculate weight normalized ratings.
3. Identify positive ideal and negative ideal solutions.
4. Calculate separation measures.

$$S_i^+ = (\sum_j (v_{ij} - v_j^+)^2)^{1/2}$$

$$S_i^- = (\sum_j (v_{ij} - v_j^-)^2)^{1/2}$$

where S_i^+ and S_i^- are L2-norm distances from the target alternative i to the best and worst conditions, respectively. ($i = 1, 2, \dots, m$)

5. Calculate similarities to positive ideal solution.

$$C_i^+ = S_i^- / (S_i^- + S_i^+), (i=1, 2, \dots, m)$$

6. Rank preference order.

Analytic Hierarchy Process (AHP)

Developed by Tomas L. Saaty in the 1970s, the analytic hierarchy process is a structured method for organizing and analyzing complex decision making problems. It is based on mathematics and psychology [35], [36]. It has been extensively studied and refined since its creation, and has been applied in many cases, such as business, government, healthcare and industry.

In general, AHP first decomposes the decision problem into a hierarchy or divides it into independent sub-problems. The elements of the hierarchy can relate to any aspect of the DM, which can be accurately measured or roughly estimated, and well understood or poorly understood. Secondly, since the hierarchy is built, the user has to systematically evaluate all elements by comparing them to one another, respecting their impact or importance on an element above them in the hierarchy. These comparisons can be evaluated from concrete data or human judgments. Whatever the source of the evaluation is, AHP requires a procedure that converts these evaluations and preference to numerical values. Therefore, the entire range of the problem can be accessed, and a numerical weight can be derived for each element of the hierarchy, allowing diverse and normally incommensurable elements to be compared to one another in a static and rational way. In the last step of AHP, numerical priorities are calculated for each of the decision alternatives which represent the alternative's capability of achieving the decision goal. In addition, there are many ways to calculate those weights and priorities. Typically, right eigenvalue and mean transformation are widely applied due to the underlying mathematical concepts of them are clear and easily adopted [35], [36], [37], [38], [39].

2.3 Summary

Based on my review of previous researchers' works, there are some related contributions from other perspectives. Although the role-based evaluation and complex RBC problems are still largely unexplored, we can investigate meaningfully in this field based on their contributions.

The problems of RBC with conflict agents and agent training for sustainable groups are identified by previous works. These problems are systematically clarified by Haibin Zhu, but the solutions for these problems are not ideal as the inefficiency of the algorithms in some cases. The statements of these problems are defined in Chapter 3, and my solutions and simulations of these solutions are demonstrated in Chapter 4.

For role-based evaluation, the evaluation methods may adapt the MADM algorithms. It is hard to say which method is the "best" under a specified scenario in theory, and the appropriateness largely depends on the domain requirement. In Chapter 5, a practical project is described where I propose additional parameters and evaluation methods to establish the evaluation model for the role-based evaluation. In order to show the differences between those evaluation methods, simulation tests are conducted.

3. PROBLEM STATEMENTS AND INITIAL SOLUTIONS

A clear problem statement is the first step to solving a problem which defines all the components of a problem and their relationships. It also confines a problem in bounds and formulates the interfaces with the outside of the problem. In this chapter, first I will introduce some related concepts and terminologies of role-based collaboration. Secondly, with these concepts, the problem statements for the four types of RBC problems are defined. Thirdly, the initial solutions from previous researchers are introduced.

3.1 Related concepts

The concept of role-based collaboration is the foundation of my research, and the role is the core of RBC. However, there are many kinds of role definitions from different people through different perspectives, like management, society, philosophy, etc. Thereby, in this section, I will discuss the role definitions and clarify the role concept and its status in RBC and explain the RBC concepts and the abstracted model of RBC, which is called E-CARGO.

3.1.1 What is a role?

In common usage, the term “role” derives from the theater and refers to the part played by an actor. A position represents a specific “seat” that entails certain privileges and accompanying responsibilities [45]. It is defined by Thomas and Biddle [44] as a set of prescriptions defining what the behavior of a position member should be.

According to the Oxford English Dictionary (OED) [43], a role is

- “the part or character which one has to play, undertakes, or assumes;”
- “the part played by a person in society or life;” or
- “the typical or characteristic function performed by someone or something.”

Oxford English dictionary [43] also defines a role from the behavioral and psychological view. It is “the behavior that an individual feels it appropriate to assume in adapting to any form of social interaction; the behavior considered appropriate to the interaction demanded by a particular kind of work or social position.”

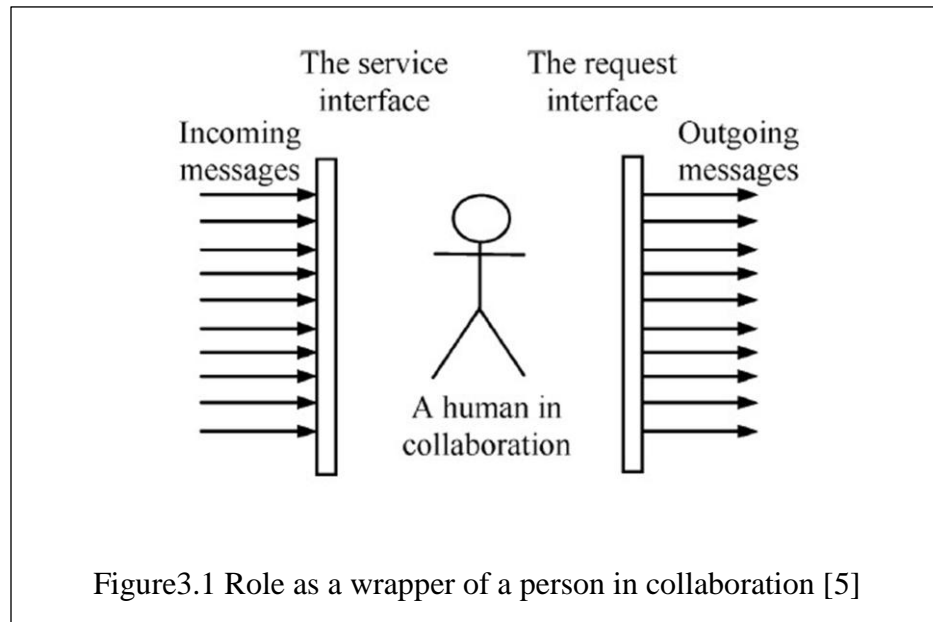
A role is defined as a regulated pattern of behavioral expectations of a person in certain situation by virtue of the person’s position in that situation [46]. A position means an institutionalized or commonly expected, accepted, and understood designation in a given social structure. For example, Chief Executive Officer (CEO) in a company, father in a family, professor in a university, where CEO, father, and professor are positions and company, family, and university are social structures [1]. Bostrom [45] defines a role as a set of expectations about behavior for a particular position within a work system. A role is a position occupied by a person in a social relationship. At this position, the person possesses special rights and takes special responsibilities. In business management, role modeling (RM) is proposed as a business engineering technique which provides a model of an organization in terms of roles, responsibilities, and collaboration among individuals and teams, and a discovery and transformation process for an enterprise, applicable in a small or large scale and a tool for reengineering and process improvement [47], [48].

Generally, we can describe rights and responsibilities in natural languages. However, it is hard to define roles clearly and exactly because they are commonly

uncertain. That is why different people in the same position can make different contributions. Role is an abstract concept. While being applied in a certain situation, a role is taken by a specific person. At this moment the behavior of this role is largely determined by the characters of the person. In a software system, we may specify roles clearly with some special considerations. The possibility comes from the exactness of computer languages, limited types and number of resources a person can access at a time, and automatic tools to retrieve information relevant to the roles. Role concepts are widely discussed and used in object-oriented systems [47], [49], [50], [51], [52], [53], [54], [55], [56], [57]. In their understanding, roles are object dependent entities that are used to confine the behavior of objects temporarily, i.e., objects play roles. However, in collaborative systems, human users play roles. A collaborative system should pay more attention to human factors. The roles in collaborative systems should be assist human users in promoting positive collaboration.

“In collaboration, a person X encounters an environment and other people who are involved. To collaborate with other people, X must know what others can do and would like to do. To work efficiently, X must know what objects can be accessed in the environment. To be cooperative, X must tell others what himself can do and what himself would like to do for them. Efficient collaboration means playing roles well. Therefore, in general, a person X has two kinds of existence: One is a server and the other a client. When X plays a role, X provides specific services and possesses specific rights to ask for services. With this common sense, a role can be defined as a person’s view of the collaborative environment. When people play a specific role, they have a specific view of the surroundings.”[5] Haibin Zhu defines a role in a collaborative environment as a

wrapper with a service interface including incoming messages and a request interface including outgoing messages as shown in Figure 3.1 [5].



In summary, roles specify not only what the system may request users to do, but also what users may ask for the system to do [5]. They include two aspects:

- Responsibilities.
- Rights

The responsibilities define what are required from the specific role. To take the role, the person needs to fully satisfy the responsibilities (this is associated with the evaluation), once the person is in that role, he get the rights to access or manipulate resources, which are also defined by the role.

3.1.2 Role-based collaboration

With a clarified definition of roles, role-based collaboration can be clearly defined. Role-based collaboration it means that people collaborate in an environment where their roles are clearly specified. Haibin Zhu defines its properties as follows [5].

- “Clear role specification: It is easy for human users to specify and understand their responsibilities and rights”.
- “Flexible role transition: It is flexible and easy for a human user to transfer from one role to another.”
- “Flexible role facilitation: It is easy for role facilitators to modify roles. Because collaborative activities are constantly evolving, even the existing roles might be required to adjust in correspondence with the development of the system.”
- “Flexible role negotiation: It is easy to negotiate a role’s specification between a human user and a role facilitator.”
- “The interactions among the collaborators are through roles.”

From these properties, it is easy to find that roles are the key media for human users to interact and collaborate with each other. The users are allowed to concentrate on interacting with other roles but not human users to make the collaboration more operable. The role specification and negotiation are the major tasks to form an environment for collaboration. The specification of the interactions among roles actually defines the procedures of collaboration. Hence, role specification is fundamental and a key mechanism.

It is proposed that the procedure of role-based collaboration is as follows [5], [9].

- Step 1) Negotiate roles. People discuss or negotiate to specify the roles relevant to collaboration. If no agreement is reached, the collaboration aborts.
- Step 2) Assign roles. Every person is assigned one or more role. If no agreement is reached, the collaboration aborts.
- Step 3) Play roles. People work according to their roles until collaboration is successfully completed.
 - Step 3.1) Check incoming messages. People understand what they need to do at this time. The incoming messages are confined by the role responsibilities (the service interface). If there are conflicts or discontents, go to Step 1).
 - Step 3.2) Issue outgoing messages. People need to access and interact with the environment by sending messages, or asking for others' services to provide their services. If there are no incoming messages, people may think and issue messages as they want. The messages are confined by their role's rights (the request interface). If there are conflicts or discontents, go to Step 1).

Based on the above discussion, we can judge if a collaborative system is role-based or not. In fact, many traditional systems that apply role concepts cannot be called role-based collaborative systems because they only support some role views but not take roles as the central collaborative mechanisms.

Sometimes, we practice collaboration based on roles even though we do not declare it. Role-based collaboration is difficult to perform in natural environments because no role specification mechanism is available. Daily role negotiation cannot

completely remove the ambiguity. This is the major barrier to implementing role-based collaboration in the real world. Role-based systems can provide clarity and enhanced support, therefore a clear model for role-based collaboration is important for design a RBC system.

3.1.3 The E-CARGO model

The E-CARGO model (Environments, Classes, Agents, Roles, Groups, and Objects) is used to define a role-based collaboration system [5], [28]. It is proposed by Haibin Zhu [5] that a role-based collaboration system Σ is defined as a 9-tuple $\Sigma ::= \langle C, O, \mathcal{A}, \mathcal{M}, \mathcal{R}, \mathcal{E}, \mathcal{G}, s_0, \mathcal{H} \rangle$, where:

- C is a set of classes;
- O is a set of objects;
- \mathcal{A} is a set of agents;
- \mathcal{M} is a set of messages;
- \mathcal{R} is a set of roles;
- \mathcal{E} is a set of environments;
- \mathcal{G} is a set of groups;
- s_0 is the initial state of a collaborative system;
- \mathcal{H} is a set of users.

With this definition, all the components of a role-based collaborative system are well formalized and we can have a much clearer view of such systems. Actually, a well-

defined internal structure is a guarantee for a successful system design and implementation. It provides the robustness, efficiency, and correctness of the entire system and decreases the risks of modeling and implementing a RBC system. The following clarification and definitions explain the detail internal and external structure of RBC system.

The initial state s_0 is expressed by initial values of all the components C , O , \mathcal{A} , \mathcal{M} , \mathcal{R} , and \mathcal{E} , such as built-in classes, initial objects, initial agents, primitive roles, primitive messages, and primitive environments [29].

With the participation of users \mathcal{H} , such as logging into a collaborative system Σ , accessing objects of the system, sending messages through roles, forming groups in environments, Σ evolves, develops, and functions. The results of collaboration form a new state of Σ that is expressed by the conditions of C , O , \mathcal{A} , \mathcal{M} , \mathcal{E} , \mathcal{G} , and \mathcal{H} . We involve \mathcal{H} to emphasis that users might be affected by collaboration. Because without the participation of users, the system can only do what the agents can do. This is why users are an essential part of a system [5].

With a role specification mechanism, we can easily support role-based collaboration, including both easy collaboration and flexible participations. We have to specify roles, build an environment, and let users play their roles in the environment for efficient collaboration. In other words, when users want to work in a group, they should first take a role to play. They are then confined at this time by the role with special responsibilities and rights. When their current role is not satisfactory, they can ask to tune

their role or change to another role. Without a role specification mechanism, we would have no way to do role transitions and tuning easily and efficiently.

In this model, we notice that a role has no knowledge about its environments and groups, because the environments and groups are derived components based on roles. As for the role of an individual, we should recognize that s/he can only be in one place at one time and is limited in the ability to move from place to place. In the above definitions, we note that roles are defined in the systems' scope. Environments and groups are built after roles are specified. This truly reflects the reality of an organization [5], [58].

The agent concept in E-CARGO model has the potential property of autonomy. When a user *X* is logged out, the agent representing *X* will replace *X* to accept messages or send messages. When the agent class develops with more facts and knowledge, it would make a better decision in responding to incoming messages; it approaches a real agent.

The agents and roles are the media in which users operate the system and interact with others. Users can contribute to collaboration by doing the following:

- Creating an agent class that mainly helps them automatically access objects, send outgoing messages and reply to incoming messages without their direct interference;
- Negotiating roles with the role facilitator and negotiators;
- Adding, deleting, and modifying the methods of an agent class and having the agent play roles;

- Playing roles, i.e., accessing objects, sending outgoing messages, and replying to incoming messages.

To understand the importance of roles in a collaborative system, we may assume that the role is removed from the system Σ . As a result, \mathcal{E} and \mathcal{G} are also eliminated because they depend on roles. It now consists of only \mathcal{C} , \mathcal{O} , \mathcal{A} , \mathcal{M} , \mathcal{S}_0 and \mathcal{H} . Thereby Σ becomes a traditional collaborative system. We lose the mechanism to organize collaboration and the media of interactions between the users and other components of the system. This is why traditional computer-supported cooperative work (CSCW) systems always argue about their multiple user interface design, because the interfaces are totally dependent on the implementations. Based on the E-CARGO model, the interfaces of a system are totally determined by the roles.

As for workflows, they can be formed by exchanging messages among roles. Workflows are defined by roles. Agents are designed to play roles to accomplish the tasks specified by a workflow. Here is an example from Haibin Zhu's previous work [5], [28], [59]. Suppose that we have roles R_1 with incoming messages m_{i11} and m_{i12} and outgoing messages m_{o11} and m_{o12} ; R_2 with m_{i21} , m_{i22} , m_{o21} , and m_{o22} ; and R_3 with m_{i31} , m_{i32} , m_{o31} , and m_{o32} . If m_{o11} matches m_{i22} , m_{o21} matches m_{i31} , and m_{o21} is required to respond to m_{i22} , we have a workflow:

$$R_1 \bullet m_{o11} \rightarrow R_2 \bullet m_{i22} \rightarrow R_2 \bullet m_{o21} \rightarrow R_3 \bullet m_{i31}.$$

Haibin Zhu also described the scenario of a collaboration based on this kind of system as follows [5].

- A collaborative system built with the role mechanisms is installed on a server.
- Each user uses an interface such as a web browser to sign up. A corresponding agent is created. Only a default role is assigned to the user.
- Users log into the system on client computers with default roles.
- Some users create classes, objects, agents, roles, environments, and groups in the system.
- Users negotiate roles and may play many roles but only one at a time. They may play different roles at different times based on the requirements of collaboration.
- A user can send messages with the request (right) interfaces relevant to their roles and get messages with the service (responsibilities) interfaces.
- The agents accept the incoming messages when the users log out.
- By timely negotiation, users can transfer roles and roles can be modified or tuned by other users with specific roles based on the requirements of collaboration.
- Through the roles, users access objects and their agents, contact other agents, join groups, and contribute to the collaboration.
- The result of the collaboration is reflected by the states of objects, roles, agents, environments, groups and users in the system.

In daily life, the roles of the members in a group may change. By these changes, different members may offer different contributions to the group even with the same role. That means, in reality, one member's role is different from another's role even if they have the same role name. The definition of roles seems to restrict the creative work of the members because a role is defined by certain message patterns. However, because message patterns are versatile, it is possible for message patterns to support the creative

work of users. If the right to create objects is granted for a role, then the users who play this role can contribute all kinds of creative work. At the same time, by interactive negotiations among users, the roles' responsibilities and rights can be modified to support the creative work of the users [5], [28], [59].

Computer-Supported Cooperative Work (CSCW) systems and intelligent collaborative systems are two different fields. The former supports people to collaborate with computer systems but the latter tries to design and implement intelligent computer systems in the same way as people's collaboration. Collaborative intelligent systems hope to process incoming messages by agents themselves. However, in a CSCW system, people decide how to respond to a message.

In fact, these two kinds of systems can be combined. The E-CARGO model is at first aiming at supporting people's collaboration. With the development and the evolution of the systems built with E-CARGO and artificial intelligence (AI) technologies, a CSCW system might become more and more intelligent and could support more and more collaborative activities. This will become true after a role engine is built that automatically distributes messages to agents that are playing the role relevant to the messages and then reply to the requesting agents.

To formally state the related concepts, the applied notations require some initial clarification. Haibin Zhu defines them as, if S is a set, $|S|$ is its cardinality. If v is a variable, $v \rightarrow S$ denotes that v may take an element of S as its value. If a and b are objects, $a.b$ denotes b of a or a 's b . $\{a, b, \dots\}$ denotes a set of enumerated elements of a , b , and others. If a and b are integers, $[a, b]$ and $(a, b]$ denote the set of all the real numbers

between a and b including a and excluding a , respectively. If \mathcal{Y} is a vector, $\mathcal{Y}[i]$ denotes the element at its i th position. If \mathcal{Y} is a matrix, $\mathcal{Y}[i, j]$ denotes the element at the intersection of row i and column j in \mathcal{Y} .

Based on the notation, Haibin Zhu defines the E-CARGO model in the sense of agent evaluation and role assignment as:

Definition 1: *role* [5], [10]. A role is defined as $r ::= \langle id, \textcircled{R}, \textcircled{W} \rangle$ where,

- id is the identification of the role;
- \textcircled{R} is the requirement for agents to play r , and
- \textcircled{W} is the rights and duties for agents to play r .

Definition 2: *agent* [5], [10]. An agent is defined as $a ::= \langle id, \textcircled{Q} \rangle$ where

- id is the identification of a ; and
- \textcircled{Q} is the abilities possessed by a .

Definition 3: *environment* [5, 10]. An environment is defined as $e ::= \langle id, \mathcal{R}_e, \textcircled{S}, \mathcal{B} \rangle$ where

- id is the identification of the environment;
- \mathcal{R}_e is a finite set of roles;
- \textcircled{S} is the shared object for \mathcal{R}_e ; and

- \mathcal{B} is a finite set of tuples consisting of roles and their range, i.e., $\langle r, q \rangle$, where $r \in \mathcal{R}_e$. The role range (also called cardinalities) q is expressed by $\langle l, u \rangle$ and tells how many agents must (l) and may (u) play r in this environment.

Definition 4: *group* [5], [10]. A group is defined as $g = \langle id, e, \mathcal{A}_g, \mathcal{J} \rangle$ where

- id is the identification of the group;
- e is an environment for the group to work;
- \mathcal{A}_g is a finite set of agents; and
- \mathcal{J} is a finite set of tuples consisting of agents and roles, i.e., $\mathcal{J} = \{ \langle a, r \rangle \mid a \in \mathcal{A}_g, r \in \mathcal{R}_e \}$.

Definition 5: *role assignment* [5], [10]. For a group g , a tuple $\langle a, r \rangle$ of $g.\mathcal{J}$ is called a *role assignment*, also called *agent assignment*.

In formalizing role assignment problems, only agents and roles are emphasized. In the following discussions, current agents or roles are our focus and environments and groups are simplified into vectors and matrices, respectively. In describing the problems, $m (= |\mathcal{A}|)$ expresses the size of the agent set \mathcal{A} and $n (= |\mathcal{R}|)$ the size of the role set \mathcal{R} .

Definition 6: *role range vector* [5], [10]. A role range vector is a vector of the lower ranges of roles in environment e of group g . Suppose that roles in $g.e$ are numbered as j ($0 \leq j \leq n-1$) and $\mathcal{B}[j]$ means the tuple for role j , then $L[j] = g.e.\mathcal{B}[j].q.l$. The role range vector is denoted as $L[j] \in \mathcal{N}$, where \mathcal{N} is the set of natural numbers and $0 \leq j \leq n-1$.

Definition 7: *qualification matrix* Q [5], [10]. The qualification matrix is an $m \times n$ matrix $Q: \mathcal{A} \times \mathcal{R} \rightarrow [0, 1]$, where, $Q[i, j] \in [0, 1]$ expresses the qualification value of agent i for role j ($0 \leq i \leq m-1$; $0 \leq j \leq n-1$), 0 means lowest and 1 the highest.

Definition 8: *role assignment matrix* [5], [10]. A role assignment matrix is defined as an $m \times n$ matrix $T: \mathcal{A} \times \mathcal{R} \rightarrow \{0, 1\}$, where $T[i, j] = 1$ expresses that agent i is assigned to role j (i.e., $\langle a_i, r_j \rangle \in g.\mathcal{J}$) and agent i is called an *assigned agent*, while $T[i, j] = 0$ means not (i.e., $\langle a_i, r_j \rangle \notin g.\mathcal{J}$).

Definition 9: *workable role* [5], [10]. Role j is *workable* in group g if it is assigned with enough agents to play it, i.e. $\sum_{i=0}^{m-1} T[i, j] \geq L[j]$.

Definition 10: *workable role assignment matrix* [5]. A workable role assignment matrix is defined as an $m \times n$ matrix $T: \mathcal{A} \times \mathcal{R} \rightarrow \{0, 1\}$, where each role is workable, i.e., $\sum_{i=0}^{m-1} T[i, j] \geq L[j] (0 \leq j \leq n - 1)$.

Definition 11: *Group qualification* [5], [10]. The group qualification q_g of group g is defined as the sum of the assigned agents' qualifications, i.e., $\sum_{i=0}^{m-1} \sum_{j=0}^{n-1} Q[i, j] \times T[i, j]$.

3.2 Problem definitions

As mentioned in the introduction, RBC collaboration problems can be categorized by different perspectives. In the real world, there are many types of RBC problems. In this thesis, I primarily discuss the following four types of RBC problems and propose improved solutions for the last two complex RBC problems:

- Basic RBC problem
- Multi-agents RBC problem
- Group role assignment problem with constraints of conflicting agents
- Agents training problem for a sustainable group

3.2.1 Basic RBC problem

Typically, basic RBC problems normally happen in such a scenario with a certain number of agents and tasks. Any agent can be assigned to perform any task, incurring some cost that may vary depending on the agent-task assignment. It is required to perform all tasks by assigning exactly one agent to each task in such a way that the total cost of the assignment is optimized.

Here is an example: Assume that there are three jobs in a restaurant, someone to wash dishes, someone to cook the food, and someone for delivery, each job needs one person to take it. There are three candidates, Jim, Tom, David, the cost for them to do different jobs are various, shown in Table 3.1, each person only can take one job at a time. The goal is to find out the assignment in which each job is assigned to a person and the total cost of the assignment is the smallest possible.

Table 3.1 Cost matrix of the restaurant

	wash dishes	cook food	delivery
Jim	\$5	\$8	\$6
Tom	\$6	\$3	\$7

While the size of the problem is small, it is not hard to find the minimum cost assignment by using the method of exhaustive search. In this case, Jim is assigned to wash dishes, Tom to cook food, and David to delivery, the total cost of this assignment is \$13 which is the smallest among all possible assignments. However, exhaustive search have to check all possibilities, assume that there are number of n jobs need to be assigned

by n candidates, due to every candidate can be assigned on any job, the number of possible assignment is:

$$n \times (n-1) \times (n-2) \times \dots \times (n-(n-2)) \times (n-(n-1)) = n!$$

While n is increasing, the time cost of the exhaustive search method to this problem is growing very fast, because its time complexity is $O(n!)$, therefore it is not suitable for calculating the assignment for large scale problems.

The basic RBC problem is the simplest collaboration problem, which can be easily defined under the E-CARGO model with a little revision. Actually, it is a special case of the E-CARGO model, the only thing we need to revise for this type of problem is Definition 6, where the role range vector is denoted as:

$$L[j] \in \{1\} \quad (0 \leq j \leq n-1), \quad \text{where } n \text{ is the number of roles.}$$

3.2.2 Multi-agents RBC problem

The multi-agents RBC problem can be totally defined by the E-CARGO model. Compared with the basic RBC problem, the major difference is the role range vector. In the basic RBC problem, each role only requires 1 agent to take, in contrast the number of required agents for each role in the multi-agents RBC problem can exceed the limit of 1, the sum of the required agent of all roles should be equal to or smaller than the number of agents in the environment, which ensures there are enough agents to be assigned.

3.2.3 Group role assignment problem with constraints of conflicting agents

More and more significant problems are unveiled with continuous research in Role-Based Collaboration (RBC) [5], [60], [61]. Group Role Assignment (GRA) [60] is a complex problem where the exhaustive-search algorithm has an exponential complexity. An efficient algorithm [60] has been developed with the application of the Hungarian algorithm, also called Kuhn-Munkres algorithm (simply K-M algorithm) [64], [65] with a polynomial complexity. However, in the real world, role assignment is not an easy process and is affected by many factors. There are many different constraints that affect the role assignment process in team work [9]. Therefore, the algorithm that applies to GRA becomes very complex if some constraints are introduced [18], [62]. For example, consider the constraint of conflict agents in the GRA problem. To clarify, the conflict here means that two agents cannot work together. Therefore, the conflicts should be resolved in the assignment matrix. However, when we resolve the conflicts at different levels, the assignment matrices might be different. Normally, the conflicts can be considered at role level or group level. If we consider the conflict at role level, in the final assignment matrix, any agents with conflict cannot be assigned to any role which requires more than one agent. If we consider the conflict at the group level, any two assigned agents in the final assignment matrix cannot have conflict. Obviously, the requirements are different when we think the conflict in different level in the group role assignment problem. In this section, I will discuss and provide solutions for these two types of Group role assignment problem with constraints of conflicting agents.

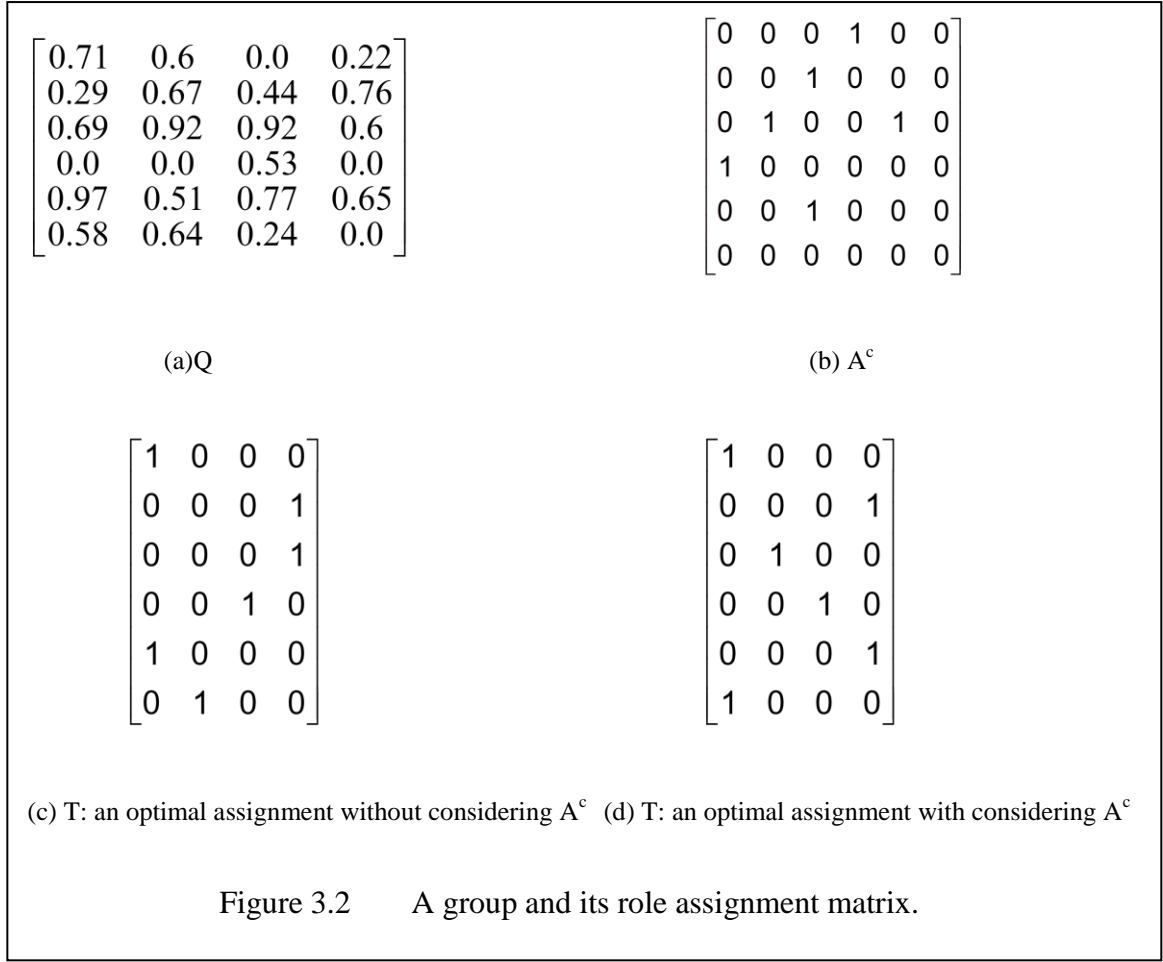
Group role assignment problem with constraints of conflicting agents (role level)

Problem statement:

Based on the E-CARGO model in Chapter 3, we may formalize the problem of group role assignment problem with constraints of conflicting agents at the role level by adding the following definitions.

Definition 12: *conflicting agents* [18], [62]. Two different agents a_i and a_j are conflicting if a_i and a_j cannot be assigned to the same role. a_i is called a conflicting agent of a_j and vice versa. An *agent conflict relation* denoted as A^c is defined as a set of tuples $\langle a_i, a_j \rangle$, where a_i, a_j are conflicting agents of each other. More exactly, $\langle a_i, a_j \rangle \in A^c \rightarrow \forall r \in \mathcal{R}, a_i, a_j \in \mathcal{A} (\neg (r=a_i.r_c \wedge r=a_j.r_c))$.

Definition 13: *conflicting agent matrix* [18], [62]. A conflicting agent matrix is defined as an $m \times m$ matrix $A^c: \mathcal{A} \times \mathcal{A} \rightarrow \{0, 1\}$, where $A^c[i, j] = 1$ expresses that agent i is conflict with agent j , while $A^c[i, j] = 0$ means not. Note: from the definitions, the conflicting agent matrix is a symmetric one along the diagonal from $[0, 0]$ to $[m-1, m-1]$, .e., $(A^c[i, j] = A^c[j, i] \wedge A^c[i, i] = 0) (i \neq j, i, j = 0, 1, \dots, m-1)$.



Definition 14: *Group Role Assignment with Constraints of Conflicting Agents (GRACCA)* (role level) [18, 62]. A GRACCA(role level) problem is to find a matrix T to form a workable group based on the requirement expressed by Q and L while T satisfies the constraint requirements provided by A^c , i.e.,

Objective:

$$\text{maximize} \{ \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} Q[i, j] \times T[i, j] \}$$

subject to:

$$\sum_{i=0}^{m-1} T[i, j] = L[j] \quad (0 \leq j \leq n-1) \quad \dots(3.1)$$

$$\sum_{j=0}^{n-1} T[i, j] \leq 1 \quad (0 \leq i \leq m-1) \quad \dots (3.2)$$

$$\sum_{i,j=0, i \neq j}^{m-1} T[i, k] \times T[j, k] \times A^c[i, j] = 0 \quad (0 \leq k \leq n-1) \quad \dots (3.3)$$

$$T[i, j] \in \{0,1\} \quad \dots (3.4)$$

Where, (3.1) expresses that T must form a workable group, called role range constraints; (3.2) expresses that one agent can only be assigned to one role at most, called the unique role constraints; (3.3) expresses that two conflicting agents cannot be assigned to the same role, called conflict constraints; and (3.4) expresses a 0-1 constraint.

For example, suppose that Q is shown as Figure 3.2(a), $L = [2, 1, 1, 2]$, A^c is shown as Figure 3.2 (b), then T in Figure 3.2(c) ($q_g = 4.21$) is not the solution because agent 1 and agent 2 are in conflict but T in Figure 3.2(d) ($q_g = 4.15$) is the solution.

Group role assignment problem with constraints of conflicting agents (group level)

Problem statement

The concepts of “conflict” in RBC with conflict agents in the group level problem and RBC with conflict agents in the role level problem are same. The major difference being that they consider the conflict at different levels. Therefore, to define a RBC with conflict agents in group level problem, we need a revised version of E-CARGO model that includes the definition of conflict (definition 12 and 13) and introduces the following definition which clarifies the group level conflict:

Definition 15: Group Role Assignment with Constraints of Conflicting Agents (GRACCA) (group level).

A GRACCA(group level) problem is to find a matrix T to form a workable group based on the requirement expressed by Q and L while T satisfies the constraint requirements provided by A^c , i.e.,

Objective:

$$\text{maximize}\{ \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} Q[i, j] \times T[i, j] \}$$

subject to:

$$\sum_{i=0}^{m-1} T[i, j] = L[j] \quad (0 \leq j \leq n-1) \quad (3.5)$$

$$\sum_{j=0}^{n-1} T[i, j] \leq 1 \quad (0 \leq i \leq m-1) \quad (3.6)$$

$$\sum_{i,j=0,i \neq j}^{m-1} T[i, j] \times T[k, l] \times A^c[i, j] = 0 \quad (0 \leq j \leq n-1, 0 \leq l \leq n-1) \quad (3.7)$$

$$T[i, j] \in \{0,1\} \quad (3.8)$$

3.2.4 Agents training problem for a sustainable group

Scenario

To understand the problem of agent training, we can consider a scenario with a soccer team. There is a group of 20 soccer players (A_0 - A_{19}) to be selected and trained to form a team. Normally, there are four types of roles in a team, goalkeeper, back fielder, middle fielder and forward. Before each season, the coach has to select and train these

players to guarantee enough players for a required formulation on the field. The coach is normally in a limited budget and time situation. He or she needs to train these players who possess different capabilities to learn to play at different positions. The capabilities are expressed by the training cost indices shown in Table 3.2. Also, the coach hopes the team can change its formation during the match. For example, the formation change plan is shown in Table 3.3.

Table 3.2 The players' training cost indices

Position Player	Goal keeper	Backward	Middlefield	Forward
A0	0.18	0.82	0.29	0.01
A1	0.35	0.8	0.58	0.35
A2	0.84	0.85	0.86	0.36
A3	0.96	0.51	0.45	0.64
A4	0.22	0.33	0.68	0.33
A5	0.18	0.75	0.47	0.73
A6	0.62	0.21	0.03	0.3
A7	0.96	0.5	0.1	0.73
A8	0.25	0.18	0.23	0.39
A9	0.56	0.35	0.8	0.62
A10	0.49	0.09	0.33	0.58
A11	0.38	0.54	0.72	0.2
A12	0.91	0.31	0.34	0.15
A13	0.85	0.34	0.43	0.18
A14	0.44	0.06	0.66	0.37

A15	0.05	0.53	0.71	0.11
A16	0.74	0.32	0.36	0.75
A17	0.08	0.06	0.83	0.78
A19	0.64	0.79	0.29	0.3
A19	0.61	0.1	0.74	0.52

(*Note: all the cost numbers are randomly generated)

Table 3.3 The players' training cost indices

Position Formation	Goal keeper	Backward	Middlefield	Forward
A	1	4	3	3
B	1	3	5	2
C	1	4	4	2
D	1	3	3	4
E	1	2	5	3
F	1	1	3	6

Problem statement

With the E-CARGO model [5], [9], [18], [30], [60], [63], a tuple of agent a and role r , i.e., $\langle a, r \rangle$ is called a role assignment (also called agent assignment). In formalizing adaptive role based collaboration problems, only agents and roles are emphasized. In the following discussions, agents or roles are the focus. Environments and groups are simplified into vectors and matrices, respectively. In the following

descriptions, $m (= |\mathcal{A}|)$ expresses the size of the agent set \mathcal{A} and $n (= |\mathcal{R}|)$ the size of the role set \mathcal{R} .

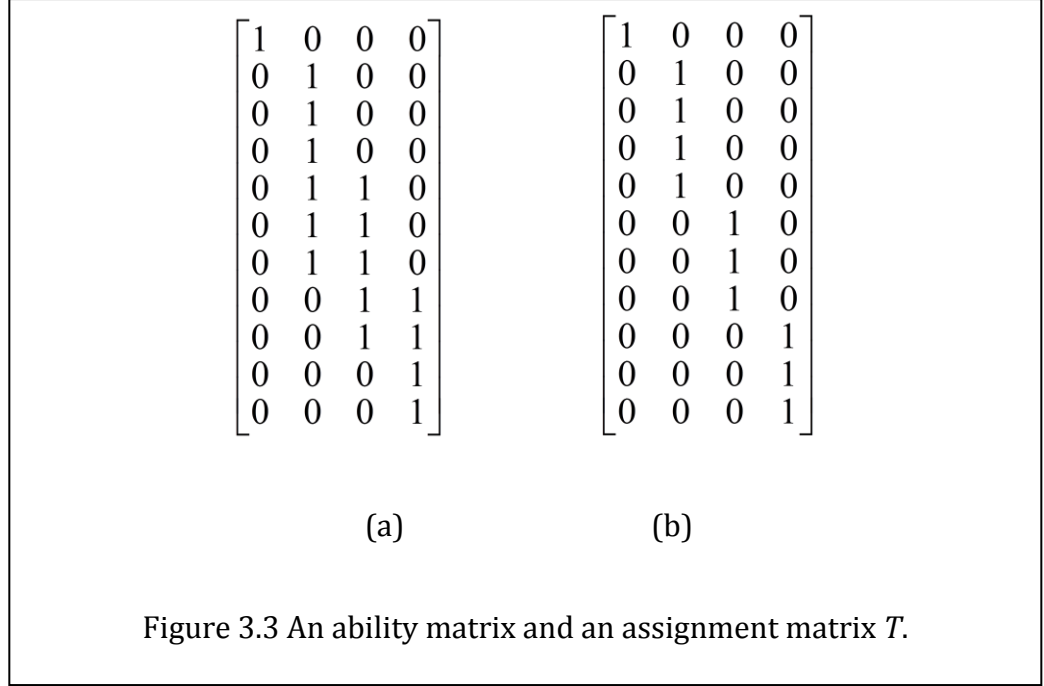
To specify the problem of sustainable group, we need to introduce a period P that is composed of t intervals, i.e., $P = \{p_0, p_1, \dots, p_{t-1}\}$.

Based on E-CARGO model, the following definitions describe an agents training problem for a sustainable group

Definition 16: *role range vector* [5], [9], [18], [30], [60], [63]. A role range vector is a vector of the lower ranges of roles in environment e of group g . The role range vector is denoted as $L[j] \in N$, where N is the set of non-negative integers, and $0 \leq j < n$. For example, $L = [1, 4, 3, 3]$ for the soccer team in Fig 3.4.

Definition 17: *ability matrix*. The ability matrix Q is an $m \times n$ matrix of values in $\{0, 1\}$, where, $Q[i, j]=1$ expresses that agent i is able to play role j and $Q[i, j]=0$ not. It is denoted as $Q[i, j] \in \{0,1\}$, $0 \leq i < m$; $0 \leq j < n$.

Definition 18: *role assignment matrix* [5], [9], [18], [30], [60], [63]. A role assignment matrix is an $m \times n$ matrix of values in $\{0, 1\}$ satisfying $\sum_{j=0}^{n-1} T[i, j] \leq 1$ ($0 \leq i < m$). If $T[i, j] = 1$, agent i is assigned to role j and agent i is called an *assigned agent*. It is denoted as $T[i, j] \in \{0,1\}$, $0 \leq i < m$; $0 \leq j < n$.



For example, Figure 3.3(a) shows an ability matrix for the soccer team in Figure 3.3 and Figure 3.3(b) shows an assignment matrix T with the formulation $L = [1, 4, 3, 3]$. The definition of T tells that *an agent can only be assigned with one role at a time*.

Definition 19: *workable role* [5], [9], [18], [30], [60], [63]. A role j ($0 \leq j < n$) is *workable* if it is assigned enough (expressed by $L[j]$) agents to play it, i.e., $\sum_{i=0}^{m-1} T[i, j] \geq L[j]$ ($0 \leq j < n$).

Definition 20: *workable group* [5], [9], [18], [30], [60], [63]. A group g is *workable* if all its roles are workable, i.e., group g expressed by T and L is *workable* if $\forall j (\sum_{i=0}^{m-1} T[i, j] \geq L[j])$ ($0 \leq j < n$).

For example, a group with the assignment matrix T shown in Figure 3.3 (b) is workable for $L = [1, 4, 3, 3]$.

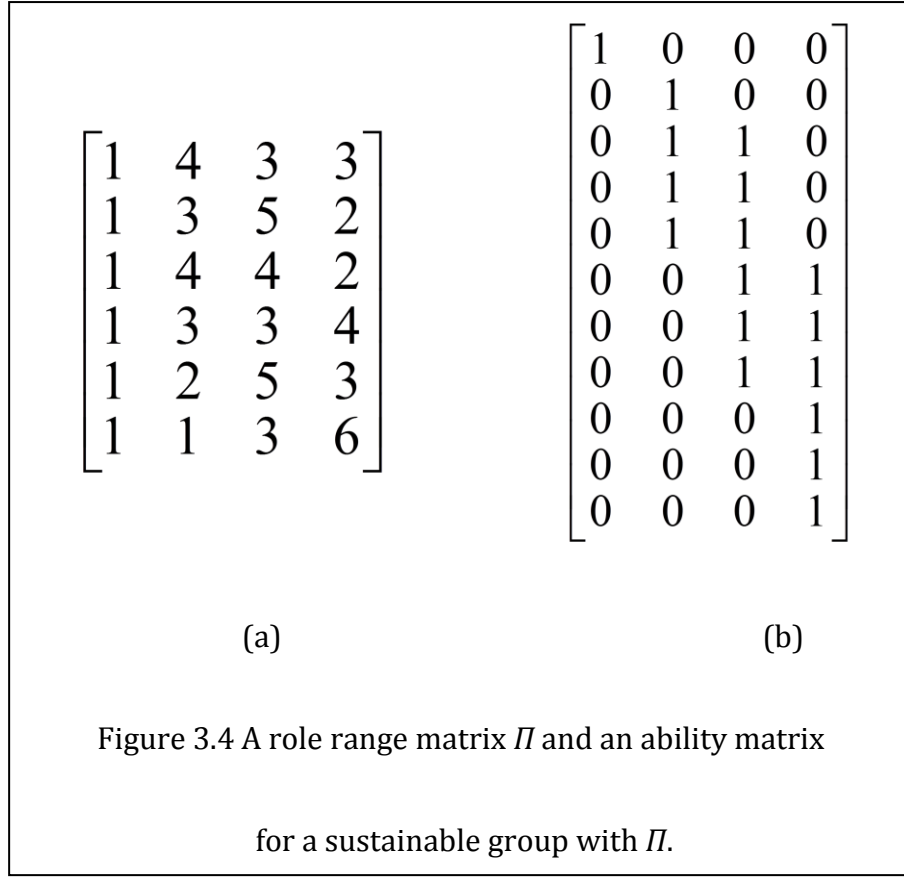
Definition 21: *role range matrix* [28], [29], [63]. A role range matrix Π is an $t \times n$ matrix, where $\Pi[k, j] \in N$, $0 \leq k < t$, $0 \leq j < n$.

In fact, Π is a vector of role range vectors, *i.e.*, $\Pi[k] = L$ at time k ($0 \leq k < t$). We use $\Pi[k]$ ($0 \leq k < t$) to express row k in matrix Π . For example, Figure 3.4 (a) is a role range matrix for the soccer team problem shown in Table 3.2 and Table 3.3.

Definition 22: *sustainable group* [28], [29], [63]. A group expressed by Q and Π is sustainable in period $P = \{p_0, p_1, \dots, p_{t-1}\}$ if the group is workable at all intervals p_k ($0 \leq k < t$), *i.e.*, at every time interval k , there exists an assignment matrix T that satisfies $\forall j (\sum_{i=0}^{m-1} T[i, j] \geq \Pi[k, j])$ ($0 \leq j < n$, $0 \leq k < t$)

For example, Figure 3.3 (a) shows an ability matrix that could not form a sustainable group for the role range matrix in Figure 3.4(a), because Figure 3.4 (a) cannot form a workable group with formulation $[1, 1, 3, 6]$. However, Figure 3.4 (b) is one that forms a sustainable group.

Obviously, if Q expresses a sustainable group, $m \geq \max \sum_{j=0}^{n-1} \Pi[k, j]$ ($0 \leq k < t$).



Problem 1: To answer whether a group expressed by an ability matrix Q is sustainable given the role range matrix Π , i.e., if there is an assignment matrix for each row of Π to satisfy:

$$\forall j(\sum_{i=0}^{m-1} T[i, j] \geq \Pi[k, j]) \quad (0 \leq j < n, 0 \leq k < t)$$

Definition 23: *simple group role assignment problem (SGRAP)* [61]. Let Q be an ability matrix and L the role range vector of group g . The SGRAP is to find a role assignment matrix T that makes g workable.

Definition 24: *training cost index matrix*[28], [29], [63]. A training cost index matrix Ω is an $m \times n$ matrix, where $\Omega[i, j] \in [0, 1]$ expresses the relative training difficulty for agent i to learn to play role j ($0 \leq i < m, 0 \leq j < n$).

Problem 2: Given m, n, Ω and Π , find a sustainable ability matrix Q that has the smallest sum of all the assigned training cost indices of Ω , i.e.,

Objective:

$$\min \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \Omega[i, j] \times Q[i, j]$$

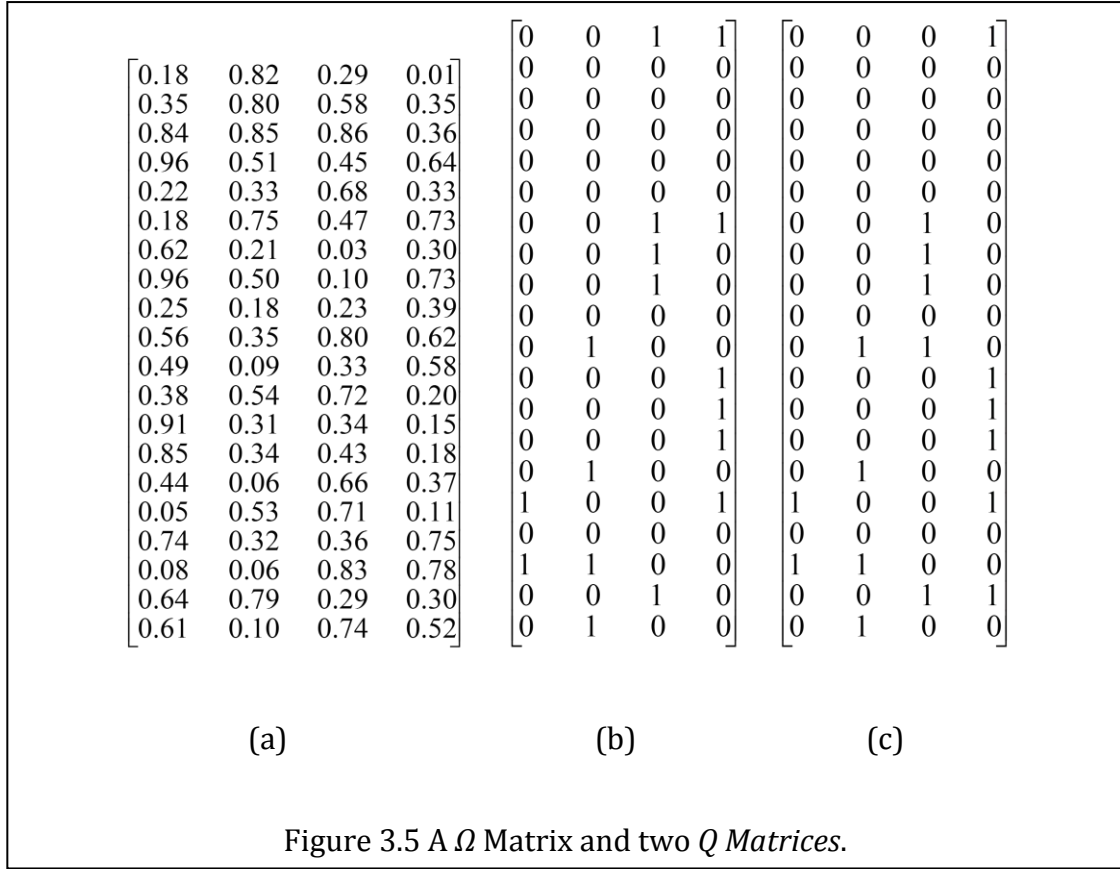
Subject to:

$$Q[i, j] = \bigvee_{k=0}^{t-1} (T_k[i, j] = 1)$$

$$(\text{True is 1, and False is 0, } 0 \leq k < t) \quad \dots(3.9)$$

$$\sum_{i=0}^{m-1} T_k[i, j] \geq \Pi[k, j] \quad (0 \leq k < t, 0 \leq j < n) \dots(3.10)$$

$$\sum_{j=0}^{n-1} T_k[i, j] \leq 1 \quad (0 \leq k < t, 0 \leq i < m) \quad \dots(3.11)$$



3.3 Initial solutions

3.3.1 Initial solution for basic RBC problem

Based on previous research, a role assignment problem can be solved by the Hungarian algorithm that can solve the problem in polynomial time. We can directly pass the qualification matrix Q to the algorithm, and then retrieve the result assignment matrix T from it. Known as Munkres algorithm or K-M algorithm, the Hungarian algorithm is a recursive method developed and published by Harold Kuhn in 1955. The original algorithm has a time complexity of $O(n^4)$, however, due to some other independent works

(Edmonds, Karp, Tomizawa), the time complexity can be improved to $O(n^3)$. The algorithm recursively resolves the problem in 7 steps.

Assume that there is n jobs assigned to n workers, define C , an $n*n$ matrix to represent the costs of each worker on each job. To find a minimized assignment, the Hungarian algorithm takes the following steps:

- Step 0: Create an $n*n$ cost matrix in which each element represents the cost of assigning one worker to a certain job. The column indicates the jobs and the row indicates the agents. Go to Step 1.
- Step 1: For each row of the matrix, find out the minimum element and subtract it from each element in that row. Go to Step 2.
- Step 2: Find a zero in the matrix, star it if there is no starred zero in its row or column. Repeat this for each element in the matrix. Go to Step 3.
- Step 3: Cover each column which contains a starred zero. If all columns are covered, the starred zeros represent an optimized assignments set, in this case, go to DONE. Otherwise, go to Step 4.
- Step 4: Find a none-covered zero and prime it. If there is no starred zero in the row containing this primed zero, Go to Step 5. Otherwise, cover this row and uncover the column containing the starred zero. Continue this manner until there are no uncovered zeros. Go to Step 6.
- Step 5: Let Z_0 represent the uncovered primed zero found in Step 4. Let Z_1 denote the starred zero in the column of Z_0 if there is one, let Z_2 denote the primed zero in the row of Z_1 . Continue this manner until it terminates at a primed zero that has no starred zero in its column. Unstar each starred zero of

the series, star all the primed zero of the series, erase all primes and uncover all the line in the matrix. Return to Step 3.

- Step 6: Find out the smallest uncovered element, subtract it from every element of each uncovered column. Return to Step 4.

My implementation of the Hungarian algorithm is in appendix II.

3.3.2 An initial solution for multi-agents RBC problem

The original idea of solving the role assignment problem with multi-agents RBC problems is trying to transfer this problem to a basic role assignment problem and then resolve this problem with the Hungarian algorithm. The difficulty here is some of the roles require more than one agent to complete it. Therefore, in the final assignment matrix, there will be more than one assignment in the column of roles which require more than one agent. This behavior is against the Hungarian algorithm, because the Hungarian algorithm can only deal with problem requests of one agent on one role. To overcome this problem, we can duplicate the columns which need more agents [10], [60]. For example, if a certain role requires 3 agents, we can make another 2 duplicated columns in the qualification matrix, so we have 3 columns containing the same cost values.

In general, we can take the following steps to resolve the problem:

- Step 1: Transfer the original qualification matrix Q to duplicated qualification matrix, based on the role range vector. (Shown in Table 3.4 and 3.5)

- Step 2: Pass the duplicated qualification matrix to Hungarian algorithm to resolve this it. (Shown in table 3.4)
- Step 3: Get the assignment result from Hungarian algorithm, then transfer it back by combine the columns under the same role. (Shown in Table 3.6 and 3.7)

Table 3.4 Original qualification matrix and role range vector

Role Agent	A	B	C	D	E
1	0.48	0.7	0.23	0.53	0.71
2	0.65	0.58	0.13	0.1	0.41
3	0.05	0.12	0.36	0.82	0.33
4	0.28	0.85	0.58	0.49	0.49
5	0.1	0.96	0.05	0.74	0.51
6	0.67	0.19	0.62	0.34	0.18
7	0.1	0.36	0.87	0.39	0.04
8	0.53	0.24	0.11	0.06	0.17
9	0.92	0.22	0.49	0.39	0.53
10	0.71	0.16	0.44	0.03	0.92
Role range vector					
L:	2	1	2	1	1

Table 3.5 Transferred qualification matrix and matching array

Role Agent	A	A'	B	C	C'	D	E
1	0.48	0.48	0.7	0.23	0.23	0.53	0.71
2	0.65	0.65	0.58	0.13	0.13	0.1	0.41
3	0.05	0.05	0.12	0.36	0.36	0.82	0.33
4	0.28	0.28	0.85	0.58	0.58	0.49	0.49
5	0.1	0.1	0.96	0.05	0.05	0.74	0.51

6	0.67	0.67	0.19	0.62	0.62	0.34	0.18
7	0.1	0.1	0.36	0.87	0.87	0.39	0.04
8	0.53	0.53	0.24	0.11	0.11	0.06	0.17
9	0.92	0.92	0.22	0.49	0.49	0.39	0.53
10	0.71	0.71	0.16	0.44	0.44	0.03	0.92
Matching Array:							
L:	L[0]	L[0]	L[1]	L[2]	L[2]	L[3]	L[4]

Table 3.6 Transferred assignment matrix and matching array

Role Agent	A	A'	B	C	C'	D	E
1	0	0	0	0	1	0	0
2	0	0	0	1	0	0	0
3	0	1	0	0	0	0	0
4	0	0	0	0	0	0	0
5	1	0	0	0	0	0	0
6	0	0	1	0	0	0	0
7	0	0	0	0	0	0	1
8	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0
10	0	0	0	0	0	1	0
Matching Array:							
L:	L[0]	L[0]	L[1]	L[2]	L[2]	L[3]	L[4]

Table 3.7 Compressed assignment matrix and role range vector

Role Agent	A	B	C	D	E
1	0	0	1	0	0
2	0	0	1	0	0
3	1	0	0	0	0
4	0	0	0	0	0
5	1	0	0	0	0
6	0	1	0	0	0
7	0	0	0	0	1

8	0	0	0	0	0
9	0	0	0	0	0
10	0	0	0	1	0
Role range vector					
L:	2	1	2	1	1

3.3.3 An initial solution for the group role assignment problem with constraints of conflicting agents

Two initial solutions to the Group role assignment problem with constraints of conflicting agents are proposed in Haibin's previous work [18], [62]. The first solution is:

- 1) Apply the algorithm for the basic RBC problem to get one T' for each newly created Q' matrix from original Q matrix by removing conflicts;
- 2) Compare and record the largest group performance and the corresponding assignment matrix to g_{max} and T_{max} respectively; and
- 3) Complete after all the possible Q' are computed with the Hungarian algorithm.

There are $2^{s \times t}$ Q' matrices that need to be processed by the Hungarian algorithm and compared with the best results, where:

$$s = \sum_{j=0}^{n-1} \begin{cases} 1 & L[j] > 1; \\ 0 & L[j] \leq 1; \end{cases} \text{ and } t = \sum_{i=0}^{m-1} \sum_{j=i+1}^{n-1} A^c[i, j]$$

To get one Q' matrix, it is approximately $O(s \times (t+1))$. To process one Q' matrix, it is proximately $O(m^3)$, the total complexity is $O(2^{s \times t} \times s \times (t+1) \times m^3)$. It is definitely exponential and the experiments also assert this estimation [18], [62].

The second initial solution is to use recursions, i.e., recursively, assign a combination of required number of agents for each role and choose the best one among all the possible assignments. The complexity is $O(\prod_{j=0}^{n-1} \binom{L[j]}{m})$. The experiments also show the complexity is very high [18], [62].

The two initial methods above for the Group role assignment problem with constraints of conflicting agents can be applied to both RBC with conflict agents in role level and group level. From the above analysis, the previous solutions in [18], [62] are not practical. These analyses require a more efficient way to solve the RBC problem with conflict agents.

3.3.4 An initial solution of the Agents training problem for a sustainable group

For the agents training problem for a sustainable group, there are three sub-problems. From Zhu's previous work, the method for problem 1 is actually a sequence of tasks of simple group role assignment (Definition 23).

The solution to **Problem 1** can be obtained with the following two steps:

- 1) Applying at first the algorithm **SimpleAssign** (L, Q, T, m, n) [61] to each role range vector at time k ($0 \leq k < t$), i.e., $I[k]$ to check if the assignment is successful; and
- 2) If for each $I[k]$ ($0 \leq k < t$), there exists a workable assignment matrix $T[k]$, then the group expressed by Q is sustainable, otherwise not.

The complexity of algorithm **IsSustained** is $O(t \times m^3)$, because the complexity of algorithm **SimpleAssign** is $O(m^3)$.

For the **Problem 2**, an initial heuristic solution was proposed in Haibin's early work. The idea of the solution is applying the Hungarian algorithm to cost matrix formed by all the sustainable groups [18]. However, this sustainable assignment cannot guarantee the "best assignment" because it removes some potential assignments due to the Hungarian algorithm only allowing one agent assigned to one role. From the definition of this problem, one agent could be assigned only once in one group but can be assigned multiple times in different groups. Therefore, the optimality may not be satisfied. To verify the method, Haibin Zhu also conducted a simulation test which shows some cases of this, the optimality is only 64%. It is not acceptable for many applications [18]. Therefore, we need to find a better solution.

3.4 Summary

With Haibin Zhu and other previous researchers' works, the four types of RBC related problems are clearly defined in this chapter. They also provide some initial solutions for these problems. However, the initial solutions for the Group role assignment problem with constraints of conflicting agents and the agents training problem for a sustainable groups are not ideal, due to the high time complexity or indefinite optimization degree. These defects push me to study and try to resolve these problems from other perspectives. My research and proposed solutions to these two complex RBC problems are demonstrated in Chapter 4.

4. SOLVING THE COMPLEX ROLE-BASED COLLABORATION WITH LINEAR PROGRAMMING

4.1 Express and solve the RBC problem as linear programming problem

4.1.1 Transfer RBC problem to linear programming

Linear programming is a mathematical method that determines a way to achieve the best outcome (maximum or minimum) under certain constraints. Those constraints are represented as linear relationships. Therefore, linear programming is a special case of mathematical programming. It is considered a technique for the optimization of a linear objective function which is subject to linear equality and linear inequality constraints. Linear programs are problems that can be expressed in canonical form:

$$\text{Maximize/minimize: } C^T x$$

$$\text{Subject to: } Ax \leq b$$

$$\text{And: } x \geq 0$$

Due to the features of linear programming, I start by trying to solve its RBC problems. For the RBC problem, the input normally consists of a qualification matrix (Q), an agent requirement vector (L) and other support information for complex RBC problems; the output requires the optimal outcome and its corresponding assignment matrix (T) based on the input which ensures the total outcome of the assignment is the best among

all possible assignments. The initial idea is to transfer the whole RBC problem to fit the three definition aspects of linear programming, the objective, constraints, and variable intervals.

Start from the basic RBC problem. Let us assume that there are number of m agents and number of n roles that need to be filled, each role requires only one agent. The qualification of every agent to each role is defined in matrix Q ($m \times n$). The assignment matrix is T ($m \times n$). In linear programming, we assume that every element in T is a variable, say x_{ij} ($0 \leq i < m$, $0 \leq j < n$), if x_{ij} is assigned, the value is 1, if not the value is 0. Therefore, the variable interval for every element in T is the same, $\{0, 1\}$.

Obviously, the objective of RBC is the sum of the scalar production of the Q and T . It can be expressed as maximum/minimum $Q \cdot T$.

The constraints of the basic RBC problem can be categorized into two groups. I name them as role requirement constraints and unique role constraints, respectively. The role requirement constraints are defined to meet the requirement of the role, due to that a certain number of agents is required by each role. Therefore, there are n role requirement constraints. It can be expressed as $L[j] = \sum_{i=0}^m x_{ij}$ ($0 \leq j < n$). The unique role constraints are defined to meet the definition that each agent could be only assigned on one role at the same time. Therefore, there is m unique role constraints, each of it can be expressed as $\sum_{j=0}^n x_{ij} \leq 1$ ($0 \leq i < m$).

In fact, the objective and variable intervals are easily defined and transferred from RBC problem to LP expression. Instead, the constraints definition is case dependent, they

are defined by the problem definition itself, for example, if the RBC allows an agent to be assigned multiple times at the same time, the unique role constraints need to be modified.

4.1.2 Tool for resolving linear programming

IBM ILOG CPLEX Optimization Studio (ILOG) is the developer package I chose to solve the linear programming problems. It is a bundle of optimization software produced by IBM [66]. It provides an efficient way to build up optimization models in applications for all ranges of scheduling and planning problems. With the consolidation of the integrated development environment, descriptive modeling language, and built-in tools, it supports the entire model development process.

Because ILOG performs well enough in Linear Programming (LP), I decided to try to solve those complex RBC problems by transferring them to LP problems. Because I selected Java as the programming language, we need to use the “Concert Technology” optimization interface package [66] which provides interfaces to common programming languages, such as, Java, C, or C++.

To use ILOG in Java, the packages of *ilog.concert.** and *ilog.cplex.** should be imported. To solve a typical linear programming problem by these functionalities, we should follow the following four general steps:

- Creating the LP model,
- Solving the model,
- Querying results after a solution is obtained, and
- Handling error conditions.

To solve the complex RBC problem with ILOG, we need to transfer the problem to a LP problem that is adaptable to ILOG. Actually, to define an LP problem, ILOG requires the following elements:

- Objective function coefficients;
- Constraint coefficients;
- Right-hand sides; and
- Upper and lower bounds.

4.2 Solutions for group role assignment problem with constraints of conflicting agents

4.2.1 Transfer the group role assignment problem with constraints of conflicting agents (role level) to LP and resolve with IBM ILOG.

The idea of the transfer task generally includes the following 3 steps:

- 1) Find out the four elements of data required by ILOG in the existing definitions, i.e., the role range array L , and matrices Q , A^c , and T and organize these data to define an LP problem in ILOG. In this case, matrix Q expresses the objective function coefficients, T are the variables, the upper and lower bounds of T are 1 and 0 which indicate that a role is assigned or not, respectively.

- 2) Add the objective and constraint equations.

The objective of group role assignment problem with constraints of role level conflicting agents can be expressed by the one-dimensional array forms of matrix Q and matrix T . In ILOG, we can maximize or minimize this formula based on the aim.

For the constraint equations, there are three general types of constraints in the group role assignment problem with constraints of role level conflicting agents: the role range constraints, unique role constraints and conflict constraints.

To add the constraints ILOG, we have to follow three steps:

1. Declare an expression object by calling:

IloLinearNumExpr expr = cplex.linearNumExpr();

2. Add all the terms to the object *expr* by invoking the method:

expr.addTerm(expr);

3. Depend on the situation encountered, add the constraint expression to ILOG by invoking:

cplex.addEq(expr); cplex.addLe(expr); or cplex.addGe(expr);

The role range constraints mean that each role has to be taken by a number of agents which is defined in array L . Refer to formula (3.1). Obviously, the number of role

range constraints equals to the number of roles. Figure 4.1 is the algorithm which represents the role range constraints in ILOG.

```
//Constrain type 1: Add role requirement constrains,
//the number of people assigned on each role should
//meet the requirement on that role.
//Hence, n constrains will be added.

for (int i = 0; i<n; i++)
{
    IloLinearNumExpr exprReqConstrain = cplex.linearNumExpr();
    for (int j = 0; j<m; j++)
    {
        exprReqConstrain.addTerm(1, x[i+j*n]);
    }
    cplex.addEq(exprReqConstrain, L[i]);
}
```

Figure 4.1 Transfer the requirement constraints to ILOG expressions.

The unique role constraints mean that an agent can only be assigned to one role. Refer to formula (3.2) in the statement of role assignment with role level conflict agents problem in Chapter 3. The codes in Figure 4.2 represent the unique role constraints in ILOG.

```
//Constrain type 2: unique constrains here, one person can only be
//assigned on one role at one time,
//thus there are number of 'm' constrains here need to be inserted into
//the cplex obj.
for(int i=0; i<m; i++)
{
    IloLinearNumExpr exprUniConstrain = cplex.linearNumExpr();
    for(int j = 0; j<n; j++)
    {
        exprUniConstrain.addTerm(1, x[n*i+j]);
    }
    cplex.addLe(exprUniConstrain, 1.0);
}
```

Figure 4.2 Transfer the unique task constraints to ILOG expressions.

In this problem, the conflict constraints mean if there is a conflict between two agents, these two agents cannot be assigned to the same role. Therefore, for each role, we must check all the potential conflicts to make sure there is no conflict in the assignment matrix T . Refer to formula (3.3). Figure 4.3 is the algorithm which computes the conflict constraints from given matrices T , Q and array L and add them into ILOG. Because ILOG can only solve the linear equations but cannot solve the quadratic equations, so we revise the expression of (3.3) in the algorithm in another way, we check if $A^c[i, j]$ equals to 1, then we add the constraint $T[i, k] + T[j, k] \leq 1$ to ILOG. It ensures that there is at least one zero between $T[i, k]$ and $T[j, k]$ while the corresponding item in A^c is 1.

```
//Constrain type 3: The conflict constrains.
//On each role which require more than one people, all the constrains may
//occur on that role should be added
for (int r=0; r<n; r++) // Scan the cost matrix by column
{
    if ( 1 < L[r] ) //Find out all the index of x on that column
    {
        int index[] = new int[m]; //number of person
        int indexcounter = 0;
        for(int i=0; i<m*n; i++)
        {
            if(i%n==r)
            {
                index[indexcounter]=i;
                indexcounter++;
            }
        }
        //Add conflicts constrains on that role.
        for(int i=0; i<m*m; i++) //i size of the conflict chart
        {
            int row = i/m;
            int col = i%m;
            if (1 == C[i]){
                IloLinearNumExpr conflict = cplex.linearNumExpr();
                conflict.addTerm(1, x[index[col]]);
                conflict.addTerm(1, x[index[row]]);
                cplex.addLe(conflict, 1);
            }
        }
    }
}
```

Figure 4.3 Transfer the conflict constraints to ILOG expressions.

$$\begin{bmatrix} X_{00} & X_{01} & X_{02} & X_{03} \\ X_{10} & X_{11} & X_{12} & X_{13} \\ X_{20} & X_{21} & X_{22} & X_{23} \\ X_{30} & X_{31} & X_{32} & X_{33} \\ X_{40} & X_{41} & X_{42} & X_{43} \\ X_{50} & X_{51} & X_{52} & X_{53} \end{bmatrix}$$

Figure 4.4 Assignment matrix T

Take Figure 3.2 as an example. First of all, we should initialize a CPLEX object in Java by “*IloCplex cplex = new IloCplex();*” and add our optimization objective to it. The class *IloCplex* implements the interface of Concert Technology [66] which is used for creating variables and constraints. It also provides functionality for solving mathematical programming problems and accessing solution information. Then we have to initialize the assignment matrix T (Figure 4.4).

Each X_{ij} ($0 \leq i \leq 5$ and $0 \leq j \leq 5$) is a variable which can be assigned with 1 or 0 representing if agent i is assigned to role j or not. In this case, T is declared in Java as:

$$IloIntVar[] X = cplex.intVarArray(m*n, 0, 1);$$

The first parameter indicates the size of matrix T in a one-dimensional array form (called scalar form in ILOG), the second and third parameter define the range of each variable within this matrix. Note that, ILOG does not support two-dimensional array, actually X is a one-dimensional array corresponding to T . Hence from the programming perspective, we have to match the positions of variables in X to the positions in matrix T . Obviously, our objective is to maximize/minimize the scalar production of Q in an array form Q_a and X . In this case,

$$\begin{aligned}
Q \bullet T = Q_a \bullet X = & 0.71X_{00} + 0.60X_{01} + 0.00X_{02} + 0.22X_{03} \\
& + 0.29X_{10} + 0.67X_{11} + 0.44X_{12} + 0.76X_{13} \\
& + 0.69X_{20} + 0.92X_{21} + 0.92X_{22} + 0.60X_{23} \\
& + 0.00X_{30} + 0.00X_{31} + 0.53X_{32} + 0.00X_{33} \\
& + 0.97X_{40} + 0.50X_{41} + 0.77X_{42} + 0.65X_{43} \\
& + 0.58X_{50} + 0.64X_{51} + 0.24X_{52} + 0.00X_{53}
\end{aligned}$$

To add the optimization objective, we invoke the following method in Java:

cplex.addMaximize(cplex.scalProd(X, Q_a));

The next step is to add the three types of constraints to ILOG. The role range constraints are:

$$X_{00} + X_{10} + X_{20} + X_{30} + X_{40} + X_{50} = 2$$

$$X_{01} + X_{11} + X_{21} + X_{31} + X_{41} + X_{51} = 1$$

$$X_{02} + X_{12} + X_{22} + X_{32} + X_{42} + X_{52} = 1$$

$$X_{03} + X_{13} + X_{23} + X_{33} + X_{43} + X_{53} = 2$$

The unique role constraints are represented as:

$$X_{00} + X_{01} + X_{02} + X_{03} \leq 1$$

$$X_{10} + X_{11} + X_{12} + X_{13} \leq 1$$

$$X_{20} + X_{21} + X_{22} + X_{23} \leq 1$$

$$X_{30} + X_{31} + X_{32} + X_{33} \leq 1$$

$$X_{40} + X_{41} + X_{42} + X_{43} \leq 1$$

$$X_{50} + X_{51} + X_{52} + X_{53} \leq 1$$

The conflict constraints are:

$$X_{00} + X_{30} \leq 1$$

$$X_{10} + X_{20} \leq 1$$

$$X_{20} + X_{40} \leq 1$$

$$X_{03} + X_{33} \leq 1$$

$$X_{13} + X_{23} \leq 1$$

$$X_{23} + X_{43} \leq 1$$

After transferring the role assignment with role level conflict agents problem to the LP form and adding all the constraints into ILOG, we can invoke method *cplex.solve()* to solve the problem. If the invocation succeeds, we can query the optimized objective value by method *cplex.getObjValue()* and retrieve the *X* values (matrix *T*) by *double[] val = cplex.getValues(X)*. In this case, the maximum objective retrieved is 4.15 and the value of assignment matrix *T* finally is shown in Figure 4.5 that is the same as *T* in Figure 3.2(d).

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Figure 4.5 Assignment result from ILOG solution

Performance comparison and analysis

To verify the proposed approach, I conduct comparisons between the proposed approach and the initial solutions [62]. In Chapter 3, the initial solutions are argued that they are not practical even for groups when $m = 10$, i.e., they require quite long to collect the times used by small groups. Therefore, I conduct the comparisons below to confirm these theoretical analyses. The comparison experiments are conducted on the environment as shown in Table 4.1.

Table 4.1 Configurations of the test environment

Hardware	
CPU	Intel core i5-2520M @2.5GHz \times 4 cores
MEM	6Gb DDR3@1333MHz
HDD	WD WD3200BEKT @7200 rpm
Software	

OS	windows 7 Ultimate with SP1
Eclipse	Version: Indigo Service Release 2 Build id: 20120216-1857
JDK	1.7.0_05-b05 64Bit Server VM build 23.1-b03, mixed mode

In order to compare the performance of my ILOG solution with the Hungarian algorithm solution, I create a testing program which records the processing times of these two solutions on same testing data for specified rounds. Then, I find out the maximum and minimum times, and calculate the average times. The testing data, Q , A^c and L are randomly generated in each round. The size of the problem m is pre-defined and $n = m/2$. During the test, we find that our solution with ILOG CPLEX can prove a correct result within its working range ($m < 400$ and $n = m/2$) in a reasonable time. In most cases, my solution with ILOG CPLEX is much better than the Hungarian algorithm solution. In some cases, the Hungarian algorithm solution could not provide correct results when the problem size reaches a certain scale. The reason is that the Hungarian algorithm solution is to get the assignment matrix by brutal comparing all possible assignment matrixes without conflicts. The Hungarian algorithm leads to a polynomial complexity, but the comparison cases of the problem is 2^k ($k = p \times l$), where p is the number of conflicts pairs of agents and l is the number of roles whose $L[j] > I$ ($0 \leq j \leq n-1$). Therefore, the size of the problem could be extremely large. Note that 32 is the upper bound of Java built-in integer type. The initial Hungarian solution could not produce a correct assignment matrix when the problem size approaches such a scale. In normal practices, n is very easy to exceed the limit of 32. For example, if $p=5$, and $l = 7$, than $k = 35 > 32$. Hence, the GRACCA (role

level) by the Hungarian algorithm is not practical for comparison with the newly proposed solution.

Table 4.2 Test results of ILOG solution and Hungarian solution

m	n	Conflict rate	Solution with ILOG (ms)			Exhaustive search Solution (ms)		
			Ave	Max	Min	Ave	Max	Min
10	5	0.25	12.42	260.41	4.31	201.38	20137.85	77.01
20	10	0.25	17.45	128.89	9.83	N/A ^a	N/A	N/A
40	20	0.25	69.55	439.75	41.65	N/A	N/A	N/A
10	5	0.1	6.85	98.81	3.85	312.52	797.27	88.22
20	10	0.1	15.49	106.40	6.67	N/A	N/A	N/A
40	20	0.1	40.98	328.87	24.06	N/A	N/A	N/A
10	3	0.25	7.25	92.07	4.18	22.35	136.29	9.16
20	6	0.25	15.55	125.07	6.48	N/A	N/A	N/A
40	13	0.25	49.06	227.18	28.52	N/A	N/A	N/A
10	3	0.1	7.64	93.19	3.76	17.82	112.13	2.58
20	6	0.1	11.78	102.93	4.89	N/A	N/A	N/A
40	13	0.1	29.62	165.48	12.21	N/A	N/A	N/A

Another experiment is conducted to compare the performances between my solution with ILOG and the exhaustive search solution. I initialize m with 10 and increase it by 10 after each step until m equals to 250. In each step I repeat the test for 300 rounds and in each round, Q , A^c and the role range array L are randomly generated. To compare the impact of the ratio of n/m on the performance, I form two groups of tests with the ratio of 1/2 and 1/3 respectively. To find out the influence of the conflict rate (i.e., the number of 1s in A^c divided by $m \times m$), I use the conflict rates of 10% and 25% for each group. In each round of test, I record the time cost by each random case, then find out the maximum, minimum, and the average time of that round. In this manner, I hope to find out the performance trends of the two solutions and compare the difference. Unfortunately, when we are actually running the test program, the exhaustive search solution can only provide results in a reasonable time when m equals to 10. When m grows to 20, whatever the conflict rate or the ratio of n/m is, the time is not acceptable. The running of the program is terminated after 30 minutes without obtaining a result. Finally, I get a group of only comparable results when m equals to 10. Table 4.2 shows the test results.

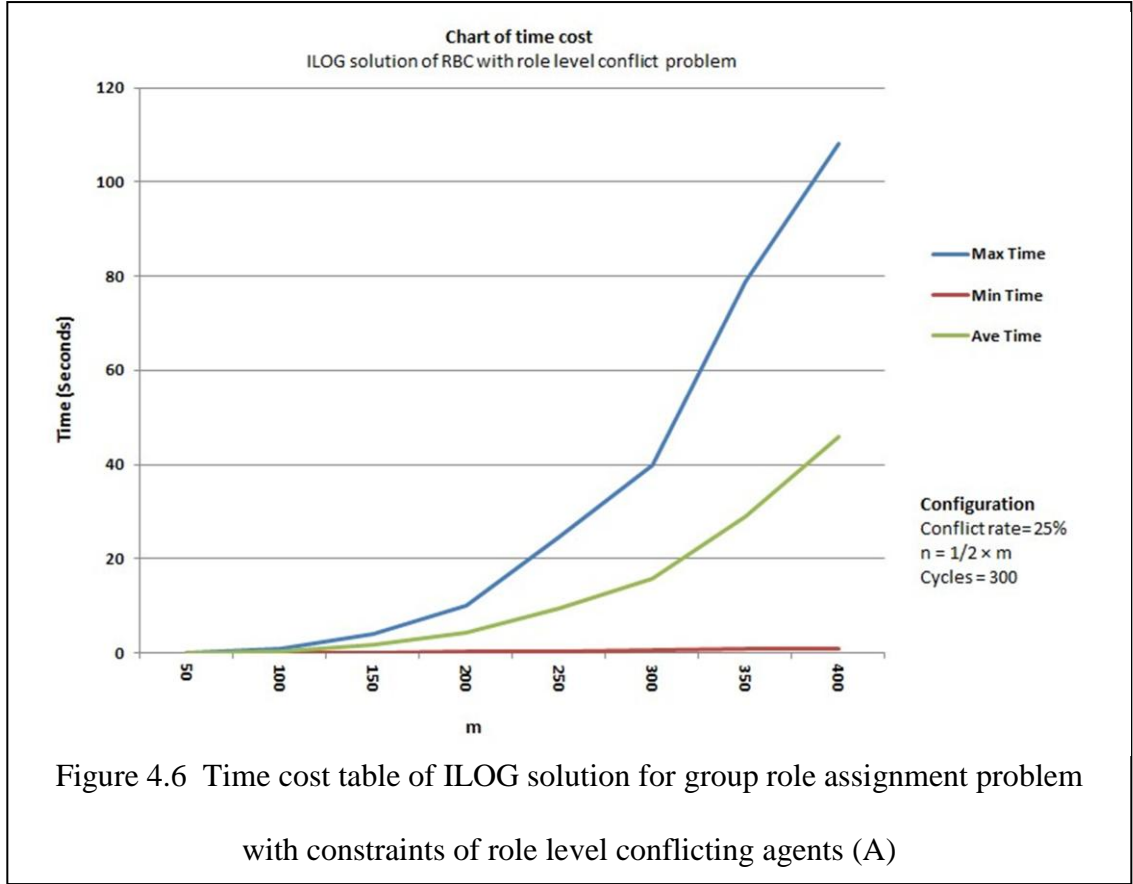
From Table 4.2, I find that the performance of solution with ILOG has an obvious relationship with the problem size. Generally speaking, we can say that the smaller the problem is, i.e., fewer conflicts and smaller Q matrix, the shorter time the solution uses. In contrast, from the limited test results, the performance of the exhaustive search solution does not have clear relationships with the conflict rate, e.g., when m equals to 10 and n equals to 5, the average time of the exhaustive search solution are 201.38ms and 312.52ms while the conflict rates are 0.25 and 0.10 respectively. However, the size of Q

affects the average time in both of the conflict rate groups. The test with a smaller n always costs less time. For instance, in the case of $m = 10$ and conflict rate = 0.25, when $n = 5$, the average time is 201.38ms; and when $n = 3$ the average time cost is 22.35ms that is much less than that of the previous one. Also, the same situation can be found in the test records when the conflict rate equals to 0.10.

To illustrate the performance, the growth trend, and find the limit of our solution with ILOG, I conducted performance experiments under different testing configurations. These experiments are tested on the same environment as shown in Table 4.1.

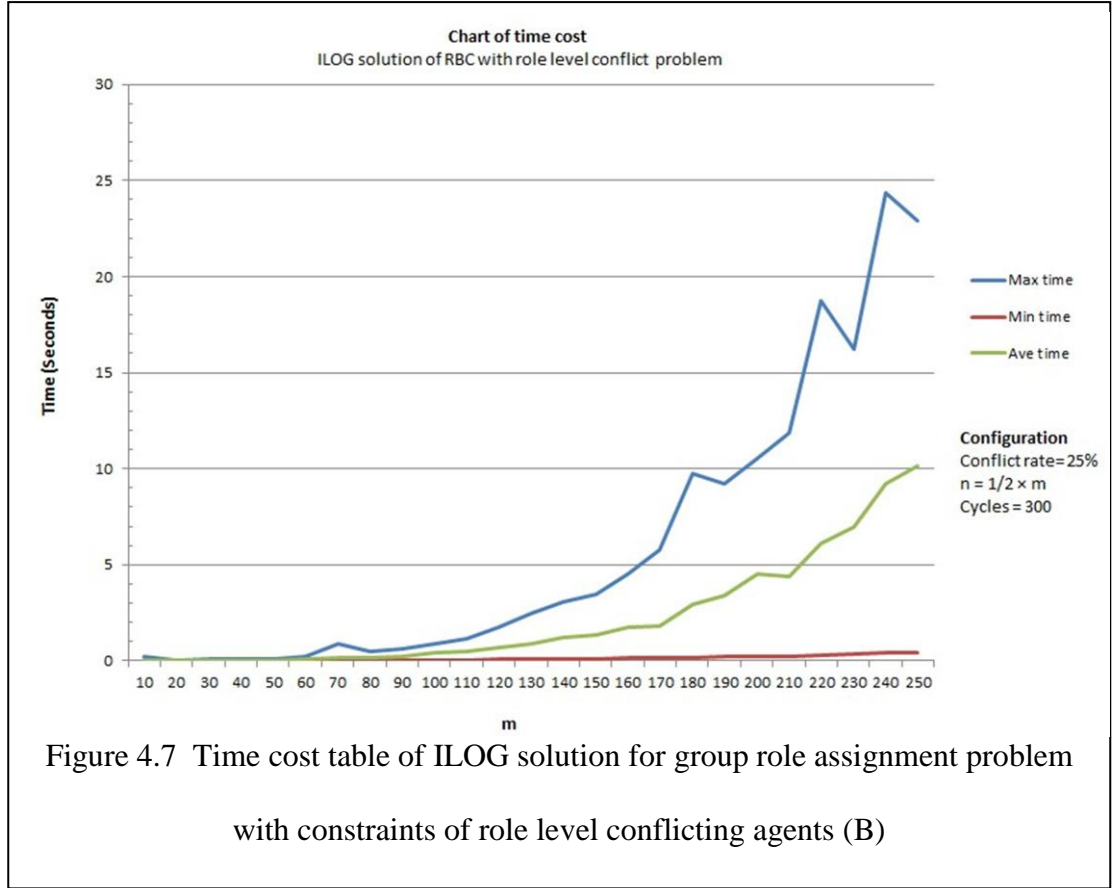
As for the limit of our solution with ILOG, I conduct tests (called limitation test) by initializing m with 100 which is increased by 50 after each step, and the upper bound of m is set to 1000. For each step, I test 300 rounds. For each round, a qualification matrix Q , a role range array L and a conflict matrix A^c with a fixed conflict rate 25% are randomly generated. The result is illustrated in Figure 4.6.

During the process of this experiment, the application encountered a “limit exceeded” error from ILOG while initializing the linear expression at step $m=450$. This is the limitation of the ILOG. Hence, to get a reasonable running time without any error for our solution, the number of variables in assignment matrix T should be controlled fewer than 100,000 which is restricted by ILOG.

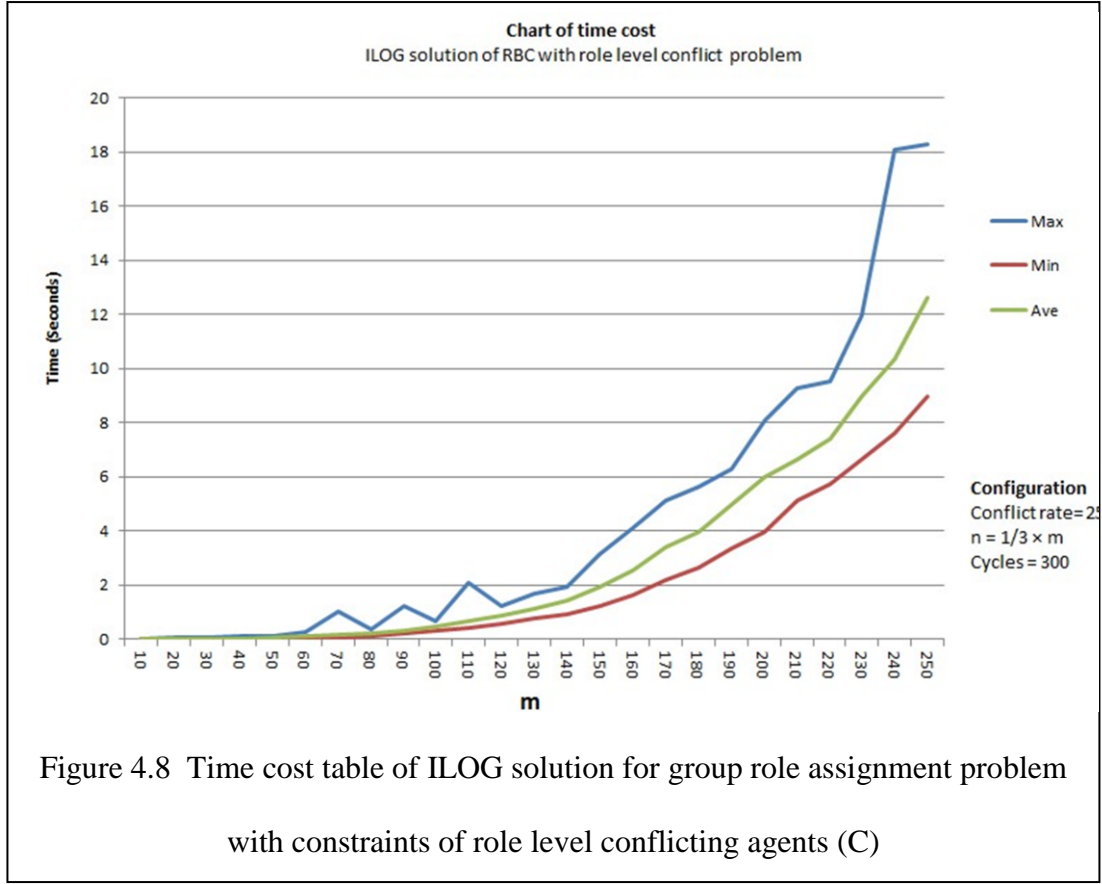


In order to understand the growth trend of the time cost, I create a testing program which randomly generates the qualification matrix Q , conflict matrix A^c , and the role range array L . The scales of Q , A^c and L are based on the number of agents (m) and roles (n) which grow after each step with a pre-defined number. In each step, I test 300 rounds to get a more accurate result. To compare the influence of all parameters, I conduct three tests: 1) regular test: test the solution with a normal setting that is called a regular configuration; 2) ratio test: change the ratio n/m and keep all other test parameters, run the test and compare it with the result of a regular configuration; and 3) conflict test: change the conflict rate but maintain all other parameters unchanged to the regular test, run the test and then compare the result with that of the regular test.

For the regular test, the conflict rate is fixed at 25% and the test range of m is initialized from 10 to 250 with a growing interval of 10 after each step, and n equals to $m/2$ in each step. In each step, I repeat the test for 300 rounds, and then find out the maximum, minimum and the average time. The result is shown in Figure 4.7.



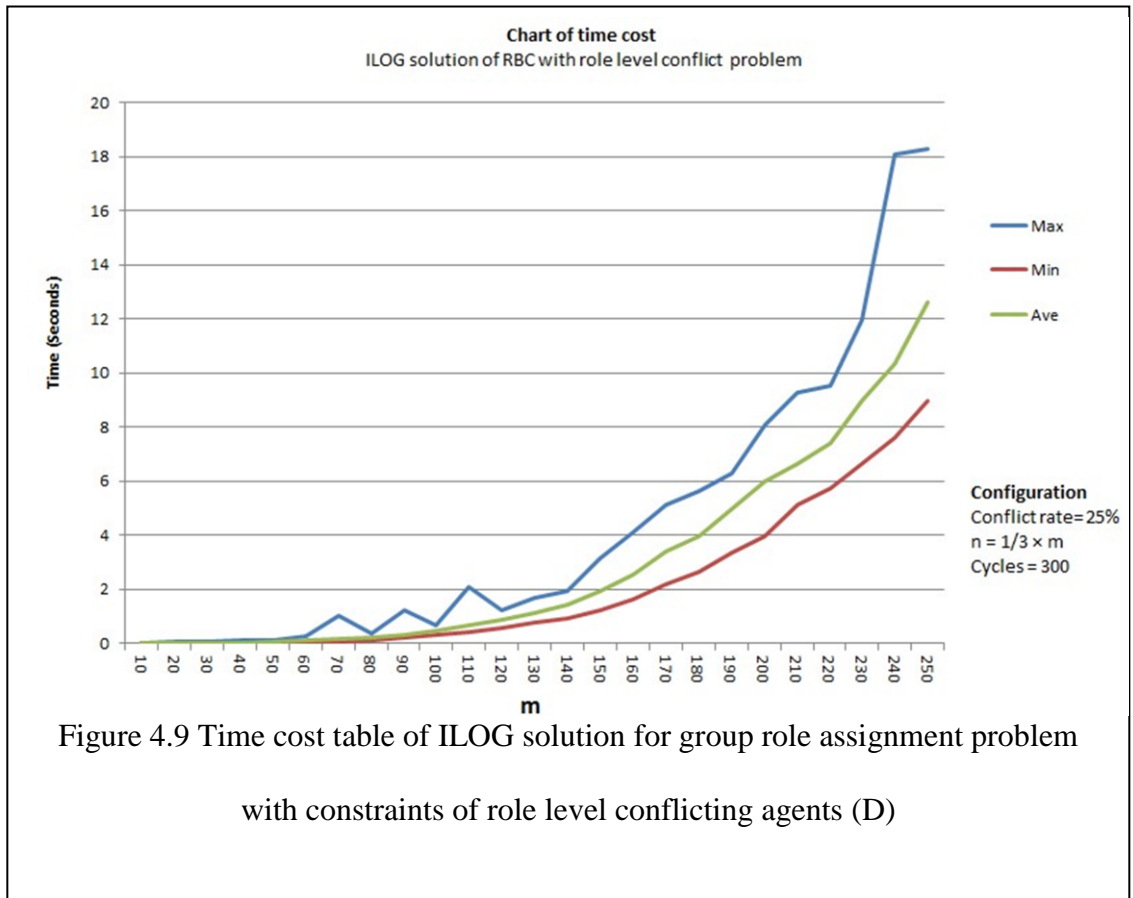
From Figure 4.7, we can easily find that the maximum and average times are apparently increased while the size of problem is growing. In contrast, the minimum time just gains a tiny steady increment from 3ms to 453ms where m is equal to 10 and 250 respectively. Also in most of cases, the value of average time is approximately equals to the mid-value of minimum and maximum time.



In the ratio test, I change the ratio n/m from $1/2$ to $1/3$, meanwhile the rest of the test parameters are kept the same as those of the regular test. The result is shown in Figure 4.8.

Compared with Figure 4.7, it is very clear that the differences between maximum, average, and minimum times are reduced in Figure 4.8. The trend of the maximum time is almost the same as that in the regular test but the values in this test are obviously smaller. Meanwhile, the trends of the minimum and average times are growing sharper than that in Figure 4.7 and the values of them are larger too. Although the size of the problem is reduced by $1/12$ compared with the regular test, there are fewer variables and constraints, but the average time increases. Therefore, we can deduce that there is no clear connection between the time and the ratio of n/m .

To understand the influence of the conflict rate, I setup a new test in which the conflict rate is changed from 25% to 10% while maintaining all other test parameters the same as those of the regular test. The result is shown in Figure 4.9.



In Figure 4.9, it is evident that the values of average times and maximum times are much smaller than those in Figure 4.7, even when a rapid increase of the maximum time occurs at $m=230$. Meanwhile, the minimum time increases faster in this case than that in the regular test. We can find that the performance of this test is better than that of the regular test due to the much smaller average time. I can also deduce that the lower the conflict rate is, the less time the program takes.

4.2.2 Transfer the group role assignment problem with constraints of conflicting agents (group level) to LP and resolve with IBM ILOG.

Similar to the process of transferring the group role assignment problem with constraints of role level conflicting agents to LP, we have to add the objective and constraint equations, too. Actually, the objectives of both problems are the same, for the problem at the group level; it also has three types of constraints: unique role constraints, role requirement constraints, and the conflict constraints. The first two types of constraints have the same concepts in role level problem; the expressions in ILOG are also the same. For the group level conflict, we consider it at a group level instead of role level; that means in the assignment matrix, any two assigned agents could not have conflict. Therefore, we have to obey the rule:

$$\sum_{i,j=0,i \neq j}^{m-1} T[i,j] \times T[k,l] \times A^c[i,j] = 0 \quad (0 \leq j \leq n-1, 0 \leq l \leq n-1)$$

To define this rule in ILOG, we have to express it in another way, because ILOG is for resolving linear programming, not quadratic equations. Hence, the method works in this way: Check every element in the conflict matrix A^c , if the value is 1 which means there is a conflict, record both its horizontal index(i) and vertical index(j), that means the agents with index i and j are conflicting. Based on the definition, the agent i and j cannot be assigned in the same group, meaning the sum of row i and j in the assignment matrix should be equal or less than 1. Therefore, the total number of conflict constraints equals to the number of conflict pair in the conflict matrix A^c . The code in Figure 4.10 shows the process of adding conflict constraints in ILOG.

```

//add conflict constraints in group level.
for (int i=0; i<m; i++)
{
    for(int j=0; j<m; j++)
    {
        if (conflictArray[i*m+j] == 1)
        {
            IloLinearNumExpr exprConflict = cplex.linearNumExpr();
            for (int col=0; col<n; col++)
            {
                exprConflict.addTerm(1, x[i*n+col]);
            }
            for(int col=0; col<n; col++)
            {
                exprConflict.addTerm(1, x[j*n+col]);
            }
            cplex.addLe(exprConflict, 1);
        }
    }
}

```

Figure 4.10 Transfer the group level conflicts to linear programming constraints

Performance of the solution for GRACCA (group level)

From the solution for group role assignment problem with constraints of group level conflicting agents, we can find that the equations of unique role constraints and role requirement constraints are same to the equations in solution for group role assignment problems with constraints of role level conflicting agents. The numbers of these two types of constraints are also equal in these two solutions. For the conflict constraints in both solutions, we can find that the expressions' complexities are similar and the numbers of conflict constraints are the same, equals to the number of conflicts in the conflict matrix. Therefore, due to the performance of linear programming algorithms is generally decided by the constraints of the linear relationships, which are the expressions of constraints in these two solutions, we can predict the performances and limits of both solutions are the

same. Thereby, for the performance of the solution for group role assignment problem with constraints of group level conflicting agents, we can refer to the performance of solution for group role assignment problems with constraints of role level conflicting agents. Figures 4.6-4.9.

4.3 Solution for agents training problem for a sustainable group

4.3.1 Transfer the agents training problem to linear programming problem and resolve with IBM ILOG

Consider Problem 2 of agents training problems for a sustainable group. Based on the statement in Chapter 3, there is a training cost index matrix Ω , we have to make an assignment (training) that can be adaptive to Π within a group and ensure the cost is minimized among all the possible assignments at the same time. Hence, the input of the problem is: m , n , Ω ($m \times n$), t , Π ($n \times t$), and the output should be an ability matrix (a special qualification matrix with values of $\{0, 1\}$ instead of $[0, 1]$ in the original Q matrix), that is why we name it Q ($m \times n$).

In general we can resolve this problem in three steps:

- Transfer the problem by expanding the inputs.
- Define the problem in ILOG and resolve it.
- Compress the result from ILOG to form the ability matrix Q .

For Step 1, we can expand the input by the following:

- 1) Horizontal duplicate the Ω for t times to form a new training cost indexes matrix Ω' . Hence, the size of Ω' is $m \times (n \times t)$.
- 2) Convert the matrix Π to array Π' by putting all the rows of Π into Π' as one row. $\Pi'[i]$ ($0 \leq i < n \times t$) indicates the number of required agents in the new training cost indexes matrix Ω' .
- 3) Based on the above two alterations, setup an new ability matrix Q' for Ω' , if $Q'[i,j]$ ($0 \leq i < m$, $0 \leq j < n \times t$) equals to 1, means agent i is trained to play role $j \% n$ (where, “%” is the integer modulo operation) in group j/n (where, “/” is the integer division operation), otherwise means not be trained for that role.

For Step 2, in order to define an agent training problem for a sustainable group by ILOG, we have to transfer the expression of the problem to linear programming based on the regulation of ILOG. To accomplish this transferring task, we need the followings:

- 1) Find out the four elements of data required by ILOG in the existing definitions. The objective coefficients can be found in Ω' , the constraints' coefficients are 1 and its right hand sides is based on the type of itself, and the bounds are 0 and 1.
- 2) Define the objective and constraint equations. The objective can be defined as:

$$\sum_{j=0}^{n \times t} Q'[i,j] \times \Omega'[i,j] \quad (0 \leq i < m)$$


```

//initialize the cplex object
IloCplex cplex = new IloCplex();

//initialize the variables array under cplex.
IloIntVar[]x = cplex.intVarArray(iM*iN*iChoice, 0, 1);

//add the optimize objective to cplex.
if(0 == flag)cplex.addMinimize(cplex.scalProd(x, dTransQ));
    else cplex.addMaximize(cplex.scalProd(x, dTransQ));

```

Figure 4.11 Declare and add the objective in ILOG

In ILOG, we can minimize or maximize the objective based on the requirement of the problem. Figure 4.11 shows the codes of defining the objective of this problem in ILOG.

There are two types of constraints in solving the training plan problem. I name the first one as the workable role constraints which means on each role, the number of required agents must be satisfied by the training assignment. Here is the expression:

$$\forall j(\sum_{i=0}^m Q'[i,j] = \Pi'[i,j]) \quad (0 \leq j < n \times t)$$

I name the second kind of constraint as unique assignment constraints which means for each role range vector, every agent can only be assigned to one role. In other words, an agent cannot be assigned to more than one role in a group at the same time. Here is the expression:

$$\sum_{j=(p \times n)}^{(p+1) \times n} T'[i,j] \leq 1 \quad (0 \leq i < m, 0 \leq p < t)$$

```

//There are two types of constraints.
//First, in each group, all the roles must be fulfilled
//according to the requirement array.
//Second, each player only can be assigned once in each group

//Constraints type 1:
for(int i=0; i<iN*iChoice; i++)
{
    IloLinearNumExpr exprRequireConstraint = cplex.linearNumExpr();
    for(int j = 0; j<iM; j++)
    {
        exprRequireConstraint.addTerm(1, x[i+j*iN*iChoice]);
    }
    cplex.addEq(exprRequireConstraint, iTransL[i]);
}
//Constrain type 2:
for (int i=0; i<iM; i++)
{
    for (int j=0; j<iChoice; j++)
    {
        IloLinearNumExpr exprUnique = cplex.linearNumExpr();
        for (int k=0; k<iN; k++)
        {
            exprUnique.addTerm(1, x[i*iChoice*iN+j*iN+k]);
        }
        cplex.addLe(exprUnique, 1);
    }
}

```

Figure 4.12 Adding constraints in ILOG.

After declaring and defining the problem, we can resolve the problem by invoking corresponding functions, which are defined within ILOG and get the status of the resolve process, if the resolving is successful, we can retrieve the result matrix Q' .

For Step 3, due to that the result matrix $Q'(m \times (n \times t))$ from ILOG indicates the result for different intervals ($I[p]$, $0 \leq p < t$) of a group, we need to compress it to a single ability matrix $Q(m \times n)$ which forms a real training plan matrix. Hence, we check all the positions in Q' and if it is 1, we set the corresponding position in Q as assigned.

```

//Solve LP
if (cplex.solve())
{
    dOptimalResult = cplex.getObjValue();
    cplex.output().println("Solution value = " + dOptimalResult);

    double[] val = cplex.getValues(x);
    int ncols = cplex.getNcols();
    dOptimalMatrix = val;

    System.out.print("\nTotal Assign Matrix\n");
    for(int i=0; i<ncols;i++)
    {
        System.out.print(dOptimalMatrix[i]+" ");
        if((i+1)%(iN*iChoice)==0) System.out.print("\n");
    }

    cplex.end();
}
else{ cplex.end(); }

```

Figure 4.13 Resolve the adaptive collaboration in ILOG.

```

//Compressing the result matrix T' to final result matrix T
for(int count=0, i=0; i<iM; i++)
{
    for(int j=0; j<iN*iChoice; j++)
    {
        iAdaptiveAssign[i][j%iChoice] = (int)dOptimalMatrix[count];
        count++;
    }
}

```

Figure 4.14 Compressing the result matrix from ILOG

to final assignment matrix T .

Performance comparison and analysis

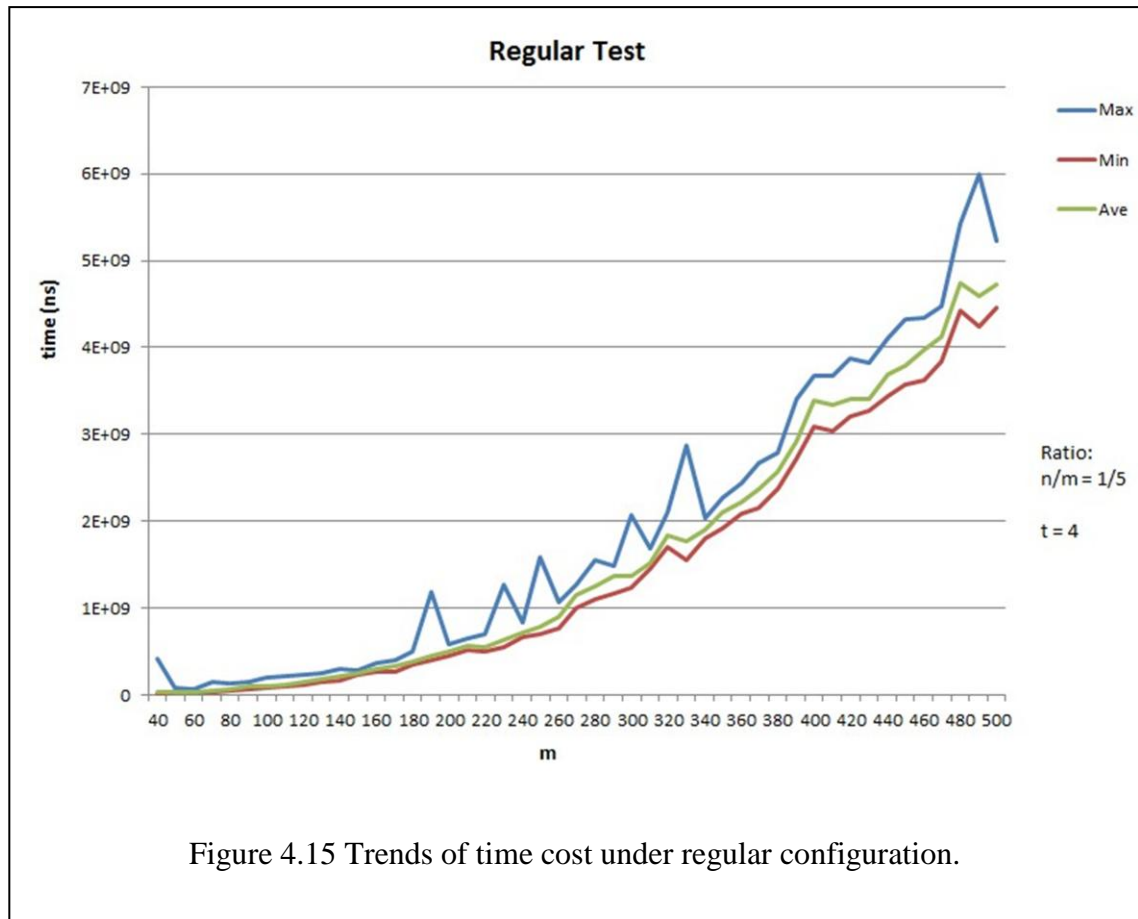
The significant factor affecting the performance of the proposed solution is the problem size that is largely influenced by m , n and t . I design a simulation program that randomly generates the original training cost index matrix Ω and role range matrix Π given m , n and t and then call the proposed method to resolve. The simulation program records the times cost by the process and gives out the maximum, minimum, and the average time cost for a specified simulation configuration. The experiments are conducted on the platform as shown in Table 4.1.

In order to understand the performance trends, we design three types of simulations:

- 1) Regular simulation. I test the solution with a normal setting called a regular configuration. Here I set the ratio of n/m with $1/5$, and the number of role range vectors 4. The simulation starts from $m = 20$, for each step we test 300 rounds and record the time cost of each round, where, m increases by 10 in each step.
- 2) Ratio simulation. I change the ratio of n/m and keep all other configurations of the regular test.
- 3) Sustainable simulation. The numbers of m and n are fixed. I change the number of role range vectors to simulate the trend of time cost.

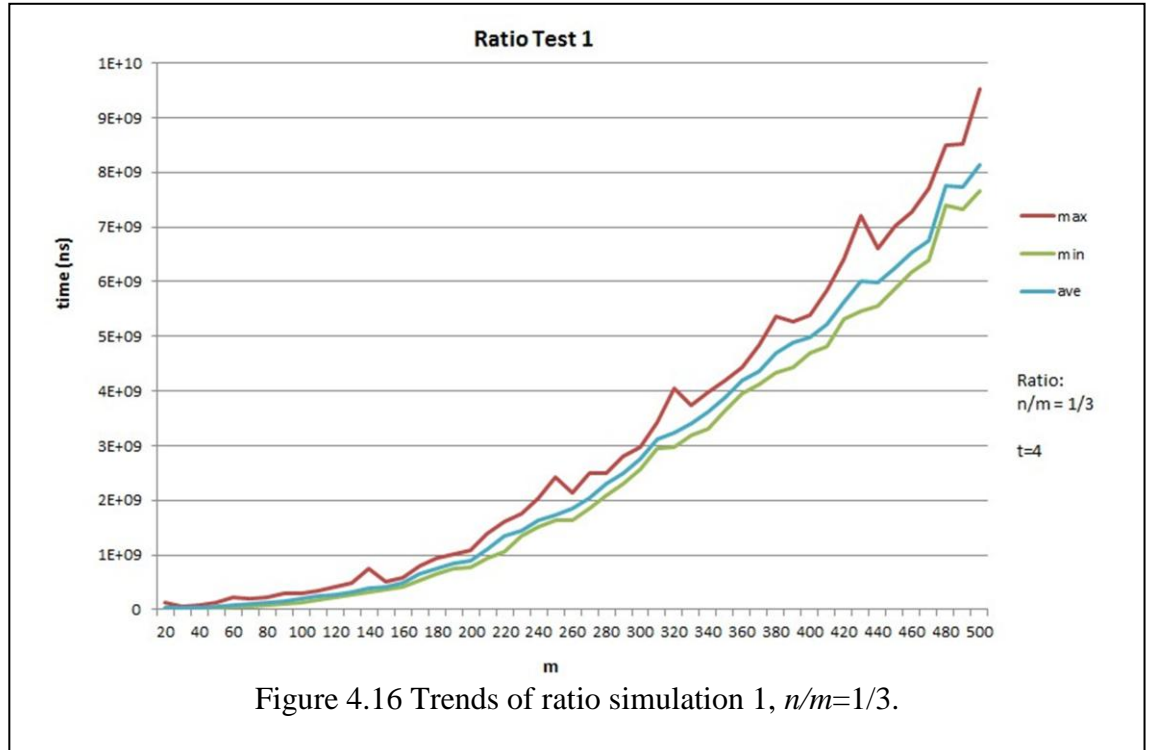
Figure 4.15 shows the trend of time cost in the regular simulation. I find that the minimum time cost is growing steadily from 0.008sec to 2.404sec while m is increasing from 20 to 500. The trend of average time cost is similar to minimum time cost, the line

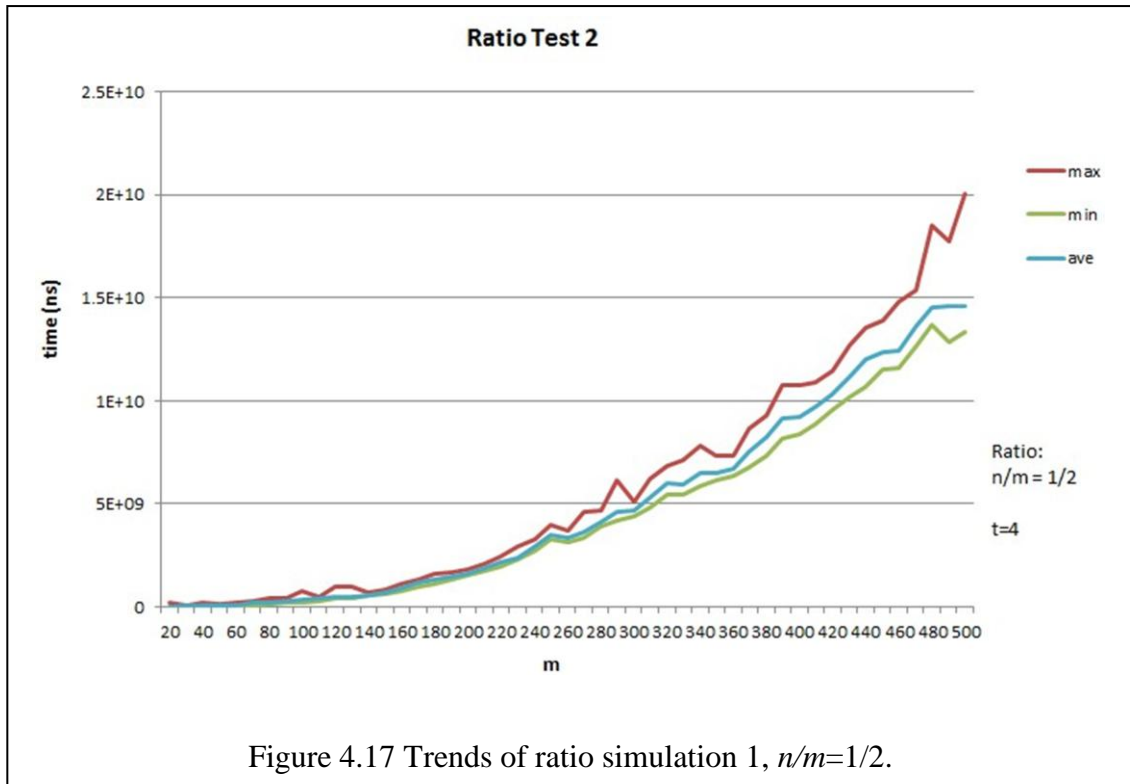
of it is higher than the minimum line but the gap between them is small. The simulation result of the maximum time cost is not stable but waving in some cases while m is growing, but in general the maximum time grows up when the size of problem is becoming larger.



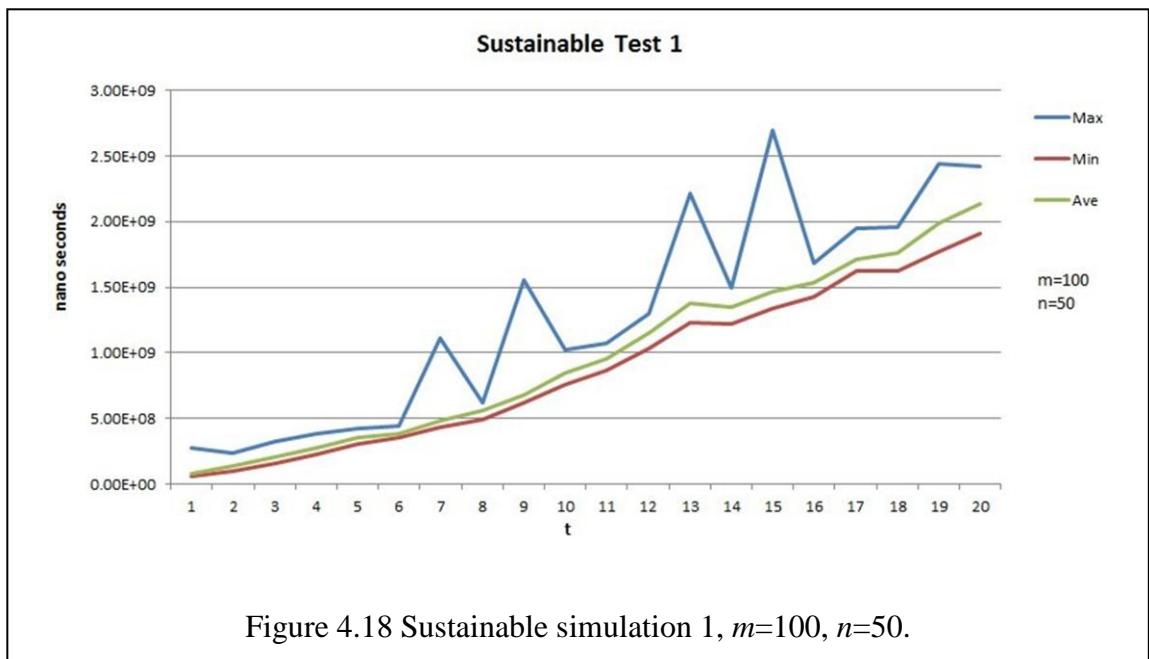
For the ratio simulation, I conducted another two tests with the ratio $1/3$ and $1/2$ respectively. The results are illustrated in Figure 4.16 and 4.17. Observing the two simulations, I found that general trends of minimum, maximum, and average time costs are growing while the size of the problem is increased from 20 to 500, the minimum and average trends are very stable, the maximum trends is waving in some cases but very stable when compared with the maximum trends in regular testing. Also, we can find that

the time cost is changing when we change the ratio of n and m , and the larger the ratio is, the greater the time cost. The reason is that in each step we have the same m but different ratios of n and m , i.e., n/m in different simulations. The larger ratio implies a larger number of n , which means a larger problem size since we have to consider more roles. And for linear programming, we need to consider more constraints. In the solution, the workable role constraints are largely influenced by this factor, because the number of these constraints equals to $n \times t$. For instance, in case of $m=500$, we have $500 \times 1/5 \times 4 = 400$ workable constraints in the regular test, but when ratio is $1/2$, we have $500 \times 1/2 \times 4 = 1000$ constraints. For the linear programming algorithm, more constraints implies more time cost, because the algorithm has to consider and satisfy more constraints at the same time. Hence, I find that the differences between the ratio simulations are increasing, when m equals 250, the average time cost is 3.49sec and 1.73sec under ratio of $1/2$ and $1/3$ respectively, but when m equals to 500, these values get to 14.60sec and 8.15sec.



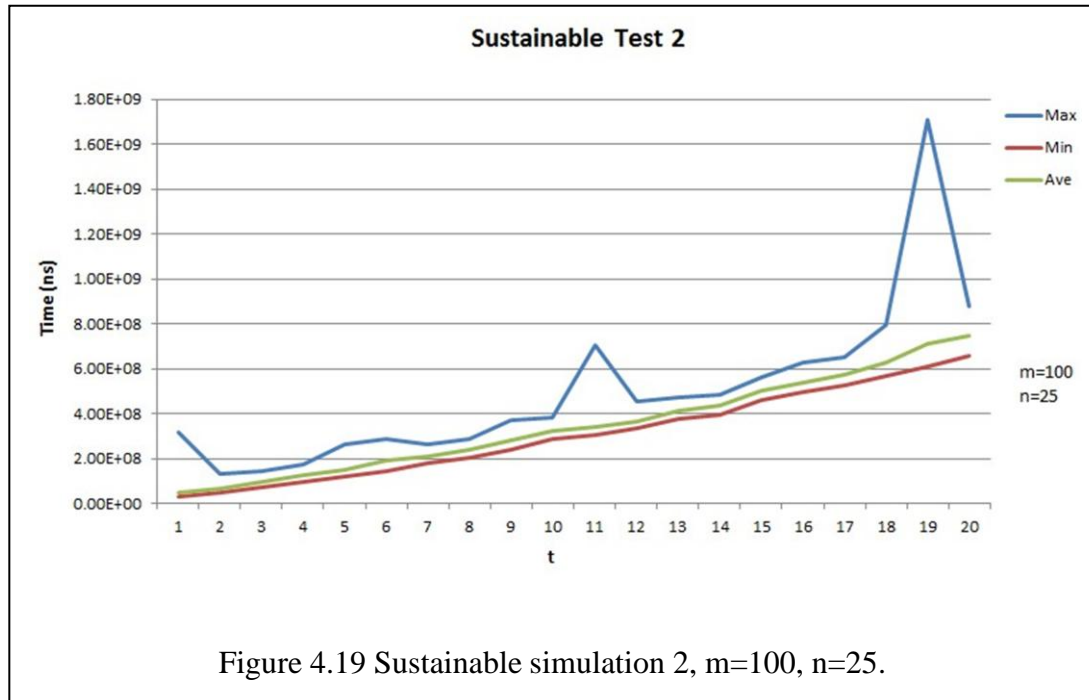


In the sustainable simulation, we keep the number of m and n fixed, and change the number of the role range vector (t) from 1 to 20 to observe the time cost trends. We conduct the simulation under two configurations, the first one is that $m = 100$ and $n = 50$.



In this configuration, we can observe that the general trends (Figure 4.18) of minimum and average time costs are increasing steadily while the number of sustainable groups is growing, the trends of maximum time cost is also growing but not stable, it encountered several waving points in cases. When $t = 10$, the average time cost is about 1sec and when $t = 20$, the average time is about 2.3sec, we can assume that in the range of 1 to 20, it is almost linear.

The second sustainable simulation is configured as $m = 100$ and $n = 25$, the trend is shown in Figure 4.19 which is almost the same as that of the previous test, except that the time cost is lower in general due to the size of the problem is reduced.



From all three types of simulations, we assert that our proposed solution is practical. The performance is vividly influenced by the number of agents, roles, and the number of changing role ranges. The problem size is clearly expressed by these factors.

5. EVALUATION IN PRACTICAL RBC SOLUTION FOR PROJECT MANAGEMENT

5.1 Introduction of the project

During my research period, I had the chance to apply my research into a project of Collective Mind Consulting Inc. (CMCI) that is an Information Technology (IT) consulting company. The company has been developing a project management software system which is called BASYS for many years. Professor Haibin Zhu of Nipissing University proposes a project that enhances their BASYS system by introducing the optimized role assignment functions into the BASYS system in order to have the best assignments of people based on their qualification towards the tasks' requirements. This proposal was supported by the NSERC Engage grant in 2013.

The co-operation with CMCI started with knowledge sharing. After we understood the state-of-art offering of both sides, we started to specify the requirements and propose initial solutions to the role assignment. The CMCI's previous solution already has the functions of the management of employee profile information. However, in order to accomplish the optimized assignment of resources, we needed to form a qualification matrix Q based on the RBC process. To form the Q , we have to evaluate the candidates. Therefore, the evaluation process is the most challenging and vital part for the project. In general, the evaluation consists of two parts: the first one is the candidates profile including their education background, professional skill level, certificates etc., that are already handled in CMCI's previous BASYS system; the other one is the definition of jobs which need to be newly designed for them, because they have no such concept in

their BASYS system. The following principles are concluded from the discussion with their Personnel Manager (PM) and Chief Technology Officer (CTO):

1. Their system will be applied in different environments, such as business and government departments. Thus, a dynamic solution which can adapt with many cases is required.
2. Human resource management and resource allocation are their greatest potential client tasks. Hence, such a scenario is their first priority consideration.
3. Based on their early work and data management, the required evaluation model is a kind of combined multiple criteria evaluation.
4. For each criterion, they would like to have a weight parameter to indicate the importance of the criteria to the general.
5. Like most other evaluation models, they would like to have a parameter called threshold to indicate if the specified criteria is considered.
6. CMCI requires different calculation methods to meet different client's preferences.

5.2 Evaluation model design for the CMCI project

To clarify the role definitions and relative concepts in the evaluation process, we can introduce the role definition based on Zhu's previous E-CARGO model [5], [30].

Based on CMCI's principles, a role is defined by a number of requirements; each requirement contains a weight and threshold. In addition, I introduce another parameter into the requirement, which it is called "*critical*". *Critical* is a boolean value indicating

whether the corresponding requirement is critical. The introduction of such a parameter is to adapt the common scenario to some requirements in a role or job that is very important. If a candidate does not have such an ability or the ability does not meet the requirement (threshold), we consider the candidate as totally not qualified for this role. For example, there is a C++ programmer job that requires an excellent skill-level with C++. Suppose that a candidate is not good at C++. However great his/her other skills are, we do not consider him/her as a potential candidate, because the C++ skill s/he has is not fit the requirement of the job.

By examining the real world human resource management cases and other relative or similar issues, I generally group the role requirements into two types: one is the requirements that can be graded, e.g., we can grade a person's C++ programming skill from 0 to 10; the other type is the requirements that only cares about whether the candidate has or not. For example, in human resource, a job usually requires that a candidate have a certain certificate, and it is meaningless to grade a person's certificate in levels. Hence, a type indicator "category" should be introduced into the role requirement which specifies the mentioned types.

Therefore a role requirement object contains the following data fields:

Table 5.1 Data structure design of role requirement

Field name	Type	Definition
Requirement Name	String	Store the name of the requirement
Requirement ID	Integer	Store the unique ID of the requirement

Weight	Double	Indicate the weight of the requirement to the role*
Threshold	Double	Indicate the lower bound of the requirement*
Critical	Boolean	Indicate whether the lack or not meet the requirement will result the fail to the role
category	Integer	Specify the type of the requirement
*when the category indicates the requirement cannot be graded, the weight and threshold are not considered.		

Then by following CMCI's principles, I designed the evaluation model for the CMCI project. The model can be divided into three levels which are shown in the Figure 5.1.

The first level is the data level, which responds to maintaining the data. The data level consists of two sub data collections: one is the role definition that contains all the requirements including weights, thresholds, etc.; the other is the candidate's information which stores the candidate's skill (ability) information. In this project, they are certificate, education information, and skill information.

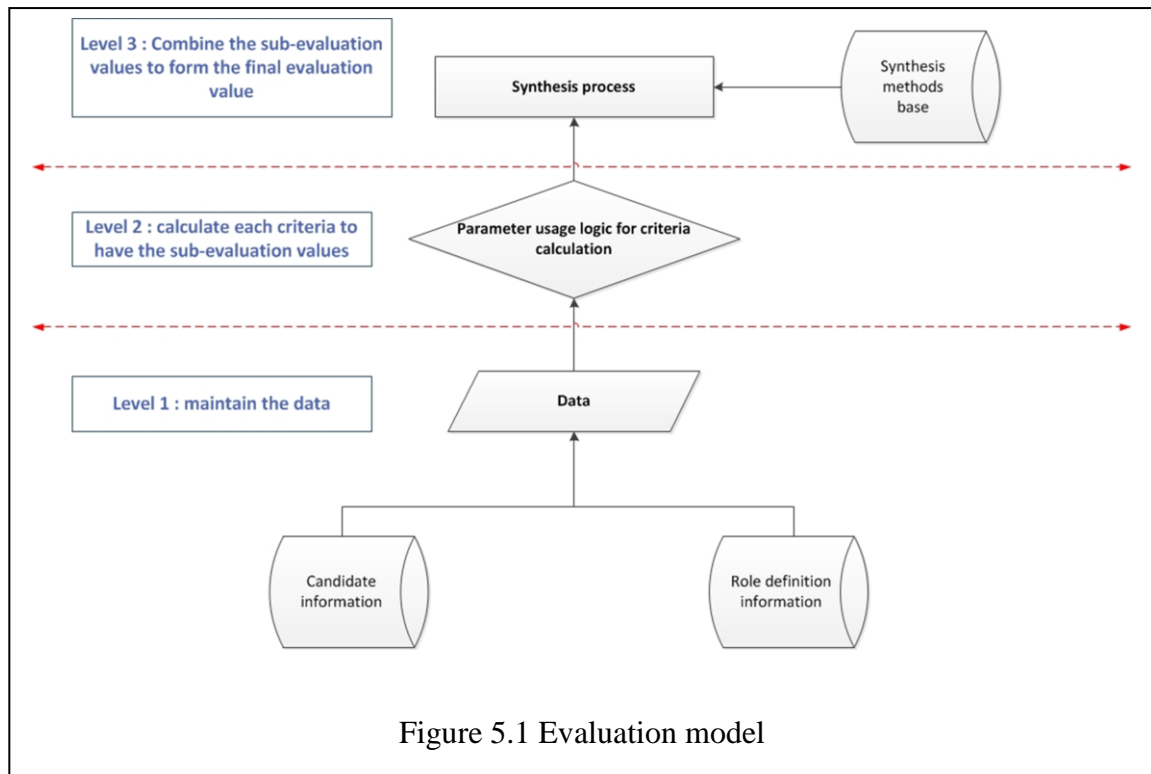
The second level is the parameter usage logic which defines the behavior of the role definition parameters (weight, threshold, critical) and defines the formula that how to calculate the sub criteria evaluation values. In normal case, the calculation logic works in this way:

- Step 1: we retrieve the category, weight, threshold, and critical values, if the category indicates that the requirement is not a grade, go to Step 2; otherwise go to Step 3.

- Step 2: we only check whether the agent (candidate) has such a required ability (skill). If it has, it passes the checking; otherwise we check if the requirement is critical. If it is, the final evaluation of this agent to the specified role is zero which means totally not qualified; if not the criteria evaluation value is zero;
- Step 3: we try to retrieve the agent's corresponding skills. If it has not the skill or its skill's level is not greater than or equal to the threshold, we consider it fails at this criterion; if this criterion is also critical, the final evaluation is failed, otherwise, we calculate the evaluation value by the formula:

$$V_{criteria} = V_{skill\ grade} \times weight$$

After level 2, if all the criteria are checked and calculated, and no critical criterion is failed, then we go to the level 3 to combine all the sub criteria evaluation together by the selected method. Actually, there could be lots of synthesis methods, I provide four methods to meet the CMCI's needs, and each method will return a decimal number between 0 and 1 which is the ratio of the agent current evaluation result with the ideal evaluation result based on the definition of the specified role.



Based on the evaluation model above, we propose the initial designs of the classes. Their relations are shown in the Figure 5.2.

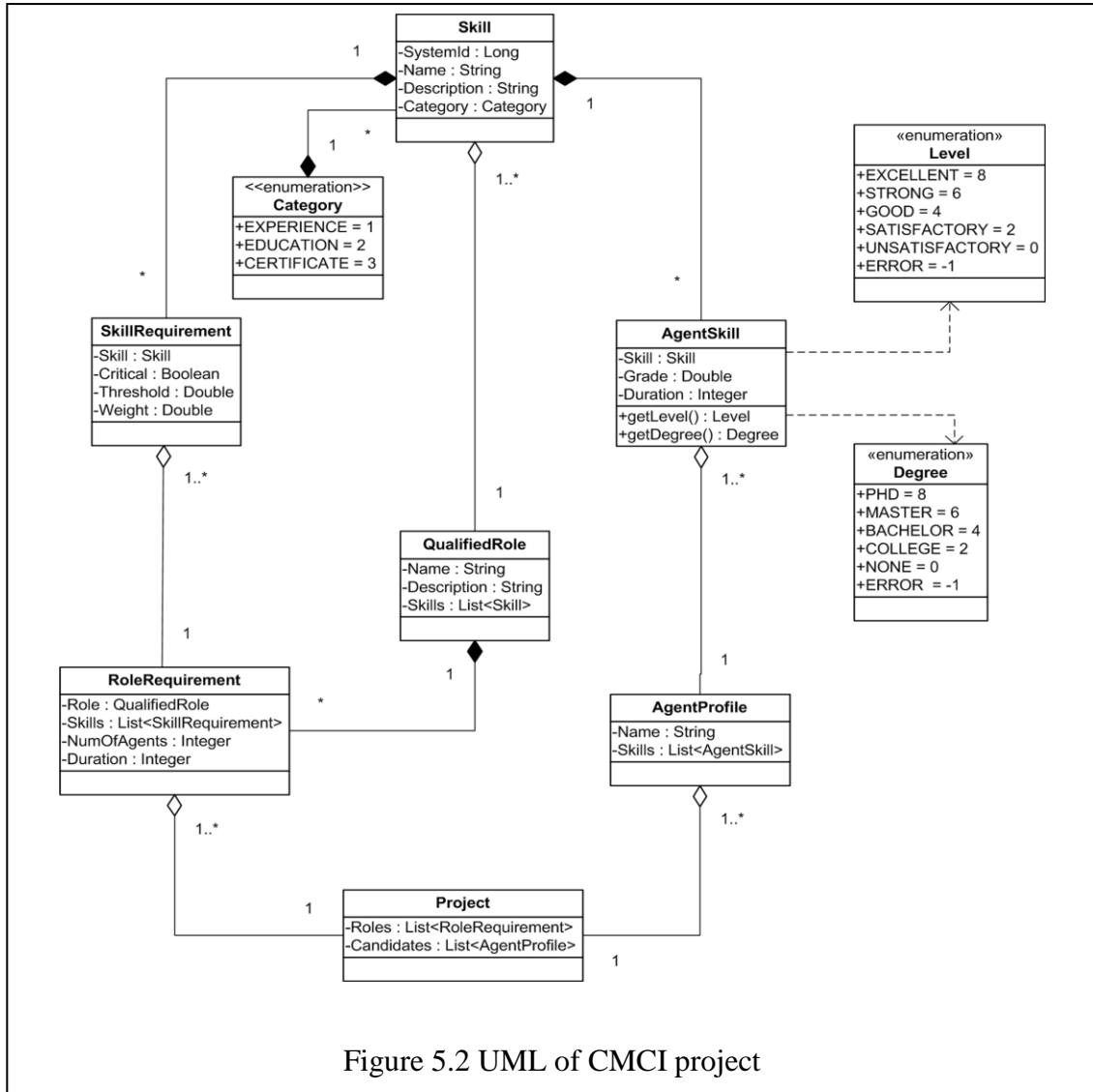


Figure 5.2 UML of CMCI project

5.3 Synthesis methods:

In the following discussions, if there are not specific explanations,

- i is the id number of the agent under evaluation, j is the id number of the corresponding role;

- n is the number evaluating ability in \mathbb{R} , S_n is the sub-evaluation of the agent on ability n ;
- S_n and $S_{n \max}$ are calculated by the simple weighting method with the parameters of the ability value, maximum ability value and ability weight.

5.3.1 Method 1: Simple Additive Weighting (SAW)

$$Q_{i,j} = S_{total} / S_{max}$$

$$S_{total} = \sum S_n$$

$$S_{max} = \sum S_{n \max} \text{ (} n \text{ is the number evaluating ability in } \mathbb{R} \text{)}$$

Features:

The simple additive weighting is a method that is close to humans' intuitions especially in case of evaluating something that can be divided into several subsets and each subset has a certain tradeoff importance degree to the final evaluation result. Hence, it is adaptive to lots of occasions. It can also be considered as a reference value compared with other evaluation methods. In addition, this method does not cost much time because its basic operations are productions and additions.

5.3.2 Method 2: Multiplicative Exponent Weighing (MEW)

$$Q_{i,j} = S_{total} / S_{max}$$

$$S_{total} = \prod S_n$$

$$S_{max} = \prod S_{n\ max} \text{ (} n \text{ is the number evaluating ability in } \textcircled{R} \text{)}$$

Features:

The multiplicative exponent weighing method uses the weight as the exponent of the ability value. Hence, a sub-evaluation result has an exponent curve in a coordinate system rather than multiplication curve of the SAW method. Also, to form the final evaluation result, the MEW method multiplies every sub-evaluation result together. Therefore, if a sub- evaluation result is 0, the final evaluation result is 0. This means that if an agent wants to be considered as a qualified one for a specified role, it must meet every ability threshold that is defined in the role requirement. Therefore, compared with SAW, MEW is more strict and sensitive. The time cost of this method is worse than that of SAW because its basic calculation is to obtain exponents and productions.

5.3.3 Method 3: Weighted Distance (WD)

$$Q_{i,j} = S_{total}/S_{max}$$

$$S_{total} = (\sum S_n^2)^{1/2}$$

$$S_{max} = (\sum S_{n\ max}^2)^{1/2} \text{ (} n \text{ is the number evaluating ability in } \textcircled{R} \text{)}$$

Features:

The weighted distance method considers that the agent is in an N-Dimension system, N equals to the number of requirements of the specified role, each sub-evaluation

value is the projection of the weighted ability value on one dimension and the final evaluation result represents the ratio of lengths of two vectors, one is the distance of the agent to the original point and the other one is the maximum distance of the agent could be. This method has a strong geometrical nature, which can be adapted to some situations which have spatial relationships among requirements. The time cost of this method is worse than SAW because its basic operations are square addition and production, besides it requires extraction of a root.

5.3.4 Method 4: Weighted Area (WA)

$$Q_{i,j} = S_{total} / S_{max}$$

$$S_{total} = \sum S_n$$

$$S_{max} = \sum S_{n\ max}$$

S_n represents the area of current point P_n , P_{n+1} and the original point. P_n is calculated by the simple weighing method which is the production of the weight (w_n) of the ability within the role and the corresponding ability value (σ_n).

$S_{n\ max}$ represents the area of an triangle formed by two current points $P_{n\ max}$, $P_{n\ max+1}$ and the original point. $P_{n\ max}$ is calculated by the simple weighing method which is the production of the weight (w_n) of the ability within the role and the maximum corresponding ability value ($\sigma_{n\ max}$).

Features:

The weighted area method regards each required ability as a vector from the original point of a 2 dimensional (2D) coordinate system, the angle between each vector is same, which is $360/n$ (n equals to the number of required abilities). Due to the WA method, agents with comprehensive sub-evaluation results will have a better final evaluation result. This method is also very sensitive because one sub-evaluation result will influence the size of its two neighbor triangles. One extreme case is: assuming that one sub-evaluation result is 0, the size of its two neighbor triangles is 0. Also, in order to form a shape in a 2D coordinate system, we need at least 3 points which means that we need 3 required abilities at least. The time cost of this method is worse than that of SAW because SAW only adds all the sub-evaluation results together, but WA needs to use the sub-evaluation results to calculate the sizes of the sub-triangles first and then add them together.

5.4 Simulation

To conduct the case study, I simulate a group which requires 5 roles, each role has 5 required abilities, in this case, each role requires the same abilities, but the requirements such as the threshold, weight, and critical values are different due to the nature of the specified role. Table 5.2 shows the definitions of those 5 roles.

We then simulate 50 agents, each agent has the 5 required abilities and the values are randomly generated. The agent information is shown in Appendix VI.

Table 5.2 The simulation of role requirements

Roles	Abilities														
	C++			Network			Operating Systems			Database			MATH		
	w	Σ	τ	w	σ	τ	w	σ	τ	w	σ	τ	w	σ	τ
C++ programmer	0.5	8	TRUE	0.1	4	TRUE	0.2	5	TRUE	0.1	6	TRUE	0.1	5	FALSE
Network Admin	0.1	3	FALSE	0.6	8	TRUE	0.1	6	FALSE	0.1	5	FALSE	0.1	5	FALSE
DBA	0.1	3	FALSE	0.1	5	FALSE	0.3	7	TRUE	0.4	7	TRUE	0.1	4	FALSE
Analyst	0	0	FALSE	0.1	5	FALSE	0.1	5	FALSE	0.3	6	TRUE	0.5	8	TRUE
IT help	0.2	4	TRUE	0.2	4	TRUE	0.2	4	TRUE	0.2	4	TRUE	0.2	4	TRUE

Based on the definition of roles and agents, we use the four evaluation methods to calculate the qualification matrices and compare the differences.

Appendix VIII shows the qualification matrix calculated by the simple additive weighting method.

Appendix IX shows the qualification matrix calculated by the multiplicative exponent weighting method.

Appendix X shows the qualification matrix calculated by the weighted distance method.

Appendix XI shows the qualification matrix calculated by the weighted area method.

6. CONCLUSION

Based on Zhu's E-CARGO model, my research contributes on these two aspects in the field of role-based collaboration:

- Re-specify the two types of complex RBC problems and proposes improved solutions for them.
- Researches different multiple criteria evaluation methods, introduces new evaluation parameters, and proposes a dynamic evaluation model to make the evaluation adaptive to real world problems.

In the aspect of complex role-based collaboration problems, group role assignment problems with constraints of conflicting agents is a complex RBC problem. The conflicts can be considered either at the role level or group level. This thesis contributes a clarification of these two types of problems and their complexities and proposes solutions for these problems. The core of my solution is transferring the group role assignment problems with conflict agents to linear programming problems which greatly improve the time costs by comparing with previous researchers' initial solutions. Simulations are also conducted which show the proposed solution obtains optimal results and performs well enough for a large set of instances of the problem ($m < 250$).

The agent training problem for a sustainable group is also a complex problem which is highly time consuming. Efficient algorithms are required to obtain an optimal result. This thesis contributes an efficient enough solution to the major problem with the hard optimization result, i.e., Problem 2 of the agent training problem for a sustainable

group, and conducts the performance experiments of the proposed solution. The experiments show that proposed solution is efficient enough for relatively large groups.

Both the solutions for these complex RBC problems can obtain the optimal assignment matrices among all possible assignment matrices and ensure the results are optimal. And these solutions are also practical which can be applied to related problems within a common problem size.

In the field of evaluation, this thesis clarifies the difficulties of connecting evaluation methods with real world requirements. In order to overcome these difficulties, I propose an evaluation model which consists of three layers: maintaining the data, calculating sub-criteria, and combining the sub-evaluation values. Each layer is adjustable and supports the layer above it. The data maintenance layer defines the role requirements and maintains the corresponding agent information. With the definition and data in the maintaining layer, the sub-criteria calculation layer could retrieve all the parameters it needs. The sub-criteria calculation layer also defines the behavior of the evaluation parameters. Based on these behaviors, it could calculate all the sub-criteria evaluation values towards a certain role. Then, the third layer combines all the sub-evaluation values by selecting a certain synthesis method. This model design is totally dynamic and expandable, and with these features, it could adapt many real world evaluation scenarios. Simply put, it is practical. In the case of the CMCI project, I introduce a new parameter called “critical”. It indicates whether an ability (skill) is vital to a role or not. By introducing this parameter, the accuracy of the evaluation in this case is improved. The parameter of “critical” may also be used for references in other evaluation environments to meet the requirements and improve the evaluation accuracy.

In general, by following Zhu's E-CARGO model for role-based collaboration, my research contributes to the assignment algorithm and evaluation in the process of group role assignment. My works greatly improve the accuracy and the time efficiency of group role assignment. The proposed solutions are practical to many real world problems under a certain scale. However, when the problem size exceeds the limit of the solution, the time efficiency is not acceptable. Thereby, there are still spaces for improving the efficiency of the algorithm. Due to the complexity of the real world scenarios, there are many other related topics that need to investigate.

7. FUTURE WORK

From this thesis, it is clear that further investigations should proceed along the following aspects:

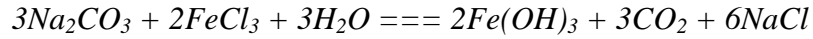
- Improve the evaluation model design, and the synthesis methods

In the practical project with CMCI, I provided four different synthesis methods to gather the sub-evaluation values together. These methods are related to the multiple criteria decision making (MCDM) algorithms. However, there are many other MCDM algorithms which can be ported to combination methods. For example, the Analytic hierarchy process (AHP) method, mentioned in Chapter 2. In AHP, the importance of each criterion is calculated automatically rather than manual judgments. Thereby the importance in AHP is rational. The difficulty of applying AHP in my evaluation model is that the process of AHP is too complicated than other methods. How to simplify the process and introduce AHP into my role-based evaluation model is a potential topic for future study.

- Study more complex role-based collaboration problems

In this thesis, I mainly focus on two types of complex role-based collaboration problems. Actually there are many other real world complex RBC problems that can be identified and defined. Even the problem in this thesis may have different behaviors under different scenarios. For example, the group role assignment problem with constraints of conflicting agents only concerns conflicts between two agents. However, the conflict may occur in a group of three agents. e.g., three

different chemicals do not react to each other, but if you put all of them together, they will react and produce new chemicals, like:



Therefore, the conflict may be defined in different group of agents. The size of a group could be 2, 3 or even more. How to solve these problems efficiently is challenging.

- Improve the efficiency of group role assignment algorithms

Although the proposed solutions in this thesis improve earlier solutions, they are still too complex for large groups, e.g., $m > 400$. There are more constraints that require considerations and more investigations are needed to conduct in this direction. Also it is valuable to investigate in the opposite views, i.e., if there are too many constraints for the proposed algorithm to work, a solution may become simple by reorganizing the teams or groups.

From the theoretical perspective, the algorithm for group role assignment problems and the study of complex RBC problems may enhance other research fields. For example, the technique of role-based collaboration can be adapted to the fields of combinatorial auctions, game theory and scheduling. The research of role-based collaboration may contribute to these related fields.

Form the application perspective, there are many software products that could be developed by referring my research in this thesis. For example, an interactive

ability evaluation tool for HR staffs, a scheduling system for outpatients in the health care sector, etc.

REFERENCES

- [1] M. Barbuceanu, T. Gray, and S. Mankovski, "Coordinating with obligations," in Proc. 2nd Int. Conf. Autonomous Agents, Minneapolis, MN, May 1998, pp. 62–69.
- [2] B. J. Biddle and E. J. Thomas, "Role Theory: Concepts and Research" New York: Krieger, 1979.
- [3] R. P. Bostrom, "Role conflict and ambiguity: Critical variables in the MIS user-designer relationship," in Proc. 17th Annu. Comput. Personnel Res.Conf., Miami, FL, 1980, pp. 88–115.
- [4] D. Hellriegel, J. W. Slocum, Jr., and R. W. Woodman, "Organizational Behavior". St. Paul, MN: West, 1983.
- [5] H. Zhu and M. Zhou, "Role-Based Collaboration and Its Kernel Mechanisms" IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART C: APPLICATIONS AND REVIEWS, VOL. 36, NO. 4, JULY 2006
- [6] Stelios H. Zanakis, Anthony Solomon, Nicole Wishart and Sandipa Dubish "Multi-attribute decision making: A simulation comparison of select methods" European Journal of Operational Research IO7 (1998) 507-529
- [7] "collaboration". Collins English Dictionary - Complete & Unabridged 11th Edition. Retrieved September 18, 2012 from CollinsDictionary.com.
- [8] Marinez-Moyano, I. J. "Exploring the Dynamics of Collaboration in Interorganizational Settings", Ch. 4, p. 83, in Schuman (Editor). Creating a Culture of Collaboration. Jossey-Bass, 2006. ISBN 0-7879-8116-8.
- [9] H. Zhu, "Fundamental Issues in the Design of a Role Engine", Proc. of the 6th International Symposium on Collaborative Technologies and Systems, Irvine, CA, USA, May 19-23, 2008, pp. 399-407.
- [10] H. Zhu, Rob Alkins, "Group role assignment" in Proc. of the ACM/IEEE International Symposium on Collaborative Technologies and Systems (CTS 09), Baltimore, MA, 2009, pp.431-439.
- [11] M. Bhardwaj and A.P. Chandrakasan, "Bounding the lifetime of sensor networks via optimal role assignments", in Proc. of Twenty-First Annual Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM 2002), New York, USA., vol. 3, June 2002, pp. 1587- 1596.
- [12] M. Dastani, V. Dignum and F. Dignum, "Role-assignment in open agent societies", Proc. of the Second International Joint Conference on Autonomous Agents and Multiagent Systems, Melbourne, Australia, 2003, pp. 489 – 496.

- [13] J.J. Odell, H. Van Dyke Parunak, S. Brueckner and J. Sauter, "Changing Roles: Dynamic Role Assignment", *Journal of Object Technology*, vol. 2, no. 5, September-October 2003, pp. 77-86.
- [14] P. Stone and M. Veloso, "Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork", *Artificial Intelligence*, vol. 110, 1999, pp. 241-273.
- [15] D. Vail, M. Veloso, "Multi-Robot Dynamic Role Assignment and Coordination through Shared Potential Fields", In A. Schultz, L. Parkera and F. Schneider (eds.), *Multi-Robot Systems*, Kluwer, 2003, pp. 87-98.
- [16] M. Nyanchama and S. Osborn, "The role graph model and conflict of interest", *ACM Transactions on Information and System Security*, vol. 2, no. 1, Feb. 1999, pp. 3-33.
- [17] Chaimowicz, L., Campos, M.F.M., Kumar, V." Dynamic role assignment for cooperative Robots". In: *Proc. of the IEEE International Conference on Robotics and Automation*, Washington, DC, USA, 1, pp. 293 – 298, IEEE (2002)
- [18] H. Zhu, "Role Assignment for an Agent Group in Consideration of Conflicts among Agents", In the *Proc. of Canadian Conference on Artificial Intelligence*, Toronto, Canada, May 28-30, 2012, L. Kosseim and D. Inkpen (Eds.): *Canadian AI 2012, LNAI vol. 7310*, pp. 267-279.
- [19] T. Dyllick, and K. Hockerts, "Beyond the business case for corporate sustainability", *Business Strategy and the Environment*, vol. 11, no. 2, 2002, pp. 130-141.
- [20] K. Kumar, and H. G. van Dissel, "Sustainable collaboration: Managing conflict and cooperation in inter-organizational Systems", *MIS Quarterly*, vol. 20, no. 3, Sep 1996, pp. 279-300.
- [21] D. L. Rulke and J. Galaskiewicz, "Distribution of Knowledge, Group Network Structure, and Group Performance", *Management Science*, Vol. 46, No. 5 (May, 2000), pp. 612-625.
- [22] L. Sandholma, "Strategic plan for sustainable excellence", *Total Quality Management & Business Excellence*, vol. 16, no. 8-9, 2005, pp. 1061-1068.
- [23] Kammer, P. J., Bolcer, G. A., Taylor, R. N., & Bergman, M. (2000). "Techniques for supporting dynamic and adaptive workflow". *Computer Supported Cooperative Work*, 9(3-4), 269-292. doi:10.1023/A:1008747109146
- [24] Faustmann, G. (2000). "Configuration for adaptation - A human-centered approach to flexible workflow enactment". *Computer Supported Cooperative Work*, 9(3-4), 413-434. doi:10.1023/A:1008719831436
- [25] Muller, R., Greiner, U., & Rahm, E. (2004). "Agentwork: a workflow system supporting rule-based workflow adaptation". *Data & Knowledge Engineering*, 51, 223-256. doi:10.1016/j.datak.2004.03.010

- [26] Berman, F., Wolski, R., Casanova, H., Cirne, W., Dail, H., & Faerman, M. (2003). "Adaptive computing on the grid using AppLeS". *IEEE Transactions on Parallel and Distributed Systems*, 14(4), 369–382. doi:10.1109/TPDS.2003.1195409
- [27] Makris, C., Panagis, Y., Sakkopoulos, E., & Tsakalidis, A. "Efficient and adaptive discovery techniques of Web services handling large data sets". *Journal of Systems and Software*, 79, 480–495. doi:10.1016/j.jss.2005.06.002
- [28] H. Zhu, M. Hou, and M.C. Zhou, "Adaptive Collaboration Based on the E-CARGO Model", *Int'l J. of Agent Technologies and Systems (IJATS)*, vol. 4, no.1, 2012, pp. 59-76.
- [29] H. Zhu, "Agent Training Plan for Sustainable Groups", in *Proc. of ACM/IEEE Conf. on Collaborative Technologies and Systems*, Denver, CO., USA, May 21-25, 2012, pp. 322-329.
- [30] H. Zhu, and M. Grenier, "Agent Evaluation for Role Assignment" in *Proc. of the IEEE 8th International Conference on Cognitive Informatics (ICCI-09)*, Hong Kong, China, 2009, pp.405-411
- [31] J. Moore, R. Inder, P. Chung, A. Macintosh and J. Stader, "Who Does What? Matching Agents to Tasks in Adaptive Workflow", *International Conference on Enterprise Information Systems 2000*.
- [32] Hwang, C.L. Yoon, K.L. "Multiple Attribute Decision Making: Methods and Applications". Springer-Verlag, New York, 1981.
- [33] Gershon, M.E., Duckstein, L. "Multiobjective approaches to river basin planning". *Journal of Water Resource Planning* 109, 13-28, 1983.
- [34] Voogd, H. "Multicriteria Evaluation for Urban and Regional Planning". Pion, London, 1983.
- [35] Saaty, Thomas L. (2008-06). "Relative Measurement and its Generalization in Decision Making: Why Pairwise Comparisons are Central in Mathematics for the Measurement of Intangible Factors - The Analytic Hierarchy/Network Process". *RACSAM (Review of the Royal spanish Academy of Sciences, Series A, Mathematics)* 102 (2): 251–318. Retrieved 2008-12-22.
- [36] Saaty, Thomas L. (2008). "Decision Making for Leaders: The Analytic Hierarchy Process for Decisions in a Complex World". Pittsburgh, Pennsylvania: RWS Publications. ISBN 0-9620317-8-X. (This book is the primary source for the sections in which it is cited.)
- [37] Saaty, Thomas L. (2010). "Principia Mathematica Decernendi: Mathematical Principles of Decision Making". Pittsburgh, Pennsylvania: RWS Publications. ISBN 978-1-888603-10-1.
- [38] Saaty, Thomas L.; Ernest H. Forman . "The Hierarchon: A Dictionary of Hierarchies". Pittsburgh, Pennsylvania: RWS Publications, 1992. ISBN 0-9620317-5-5

- [39] Saaty, Thomas L. "Fundamentals of Decision Making and Priority Theory". Pittsburgh, Pennsylvania: RWS Publications, 2001. ISBN 0-9620317-6-3.
- [40] Yoon, K. "A reconciliation among discrete compromise situations". Journal of Operational Research Society 38. pp. 277–286, 1987.
- [42] Hwang, C.L.; Lai, Y.J.; Liu, T.Y. (1993). "A new approach for multiple objective decision making". Computers and Operational Research 20: 889–899.
- [42] Yoon, K.P.; Hwang, C. (1995). "Multiple Attribute Decision Making: An Introduction". California: SAGE publications.
- [43] Oxford English Dictionary. [Online]. Available: <http://www.oed.com>
- [44] B. J. Biddle and E. J. Thomas, "Role Theory: Concepts and Research". New York: Krieger, 1979.
- [45] R. P. Bostrom, "Role conflict and ambiguity: Critical variables in the MIS user-designer relationship," in Proc. 17th Annual Compute Personnel Res.Conf., Miami, FL, 1980, pp. 88–115.
- [46] D. I. Hawkins, R. J. Best, and K. A. Coney, "Consumer Behavior". Plano, TX: Business Publications, Inc., 1983.
- [47] J. F. Patterson, "Comparing the programming demands of single-user and multi-user application," in Proc. 4th Symp. User Interface Software and Technology, Nov. 1991, pp. 87–94.
- [48] Role Modeling (RM). [Online]. Available: <http://www.rolemodeling.com>
- [49] M.Becht, T. Gurzki, J.Klarmann, andM.Muscholl, "ROPE:Role oriented programming environment for multiagent systems," in Proc. 4th IECIS Int. Conf. Cooperative Inf. Syst., Sep. 1999, pp. 325–333.
- [50] M. Dahchour, A. Pirotte, and E. Zim'anyi, "A role model and its metaclass implementation," Inf. Syst., vol. 29, no. 3, pp. 235–270, May 2004.
- [51] G. Gottlob, M. Schrefl, and B. Röck, "Extending object-oriented systems with roles," ACM Trans. Inf. Syst., vol. 14, no. 3, pp. 268–296, Jul. 1996.
- [52] B. B. Kristensen and K. Østerbye, "Roles: Conceptual abstraction theory and practical language issues," Theory Practice Object Syst., vol. 2, no. 3, pp. 143–160, 1996.
- [53] J. Murdoch and J. A. McDermid, "Modeling engineering design process with role activity diagrams," Trans. Soc. Des. Process Sci., vol. 4, no. 2, pp. 45–65, Jun. 2000.
- [54] B. Pernici, "Objects with roles," ACM SIGOIS Bull., vol. 11, no. 2–3, pp. 205–215, Mar. 1990.

- [55] D. Riehle, R. Brudermann, T. Gross, and K. U. M'atzel, "Pattern density and role modeling of an object transport service," *ACM Comput. Surv.*, vol. 32, no. 1, pp. 1–6, Mar. 2000.
- [56] F. Steimann, "On the representation of roles in object-oriented and conceptual modeling," *Data Knowl. Eng.*, vol. 35, pp. 83–106, 2000.
- [57] Workflow Management Coalition, The Workflow Reference Model.[Online]. Available: <http://www.wfmc.org/standards/docs.htm>
- [58] H. Zhu, "A role Agent Model for Collaborative Systems", *Proceedings of the 2003 International Conference on Information and Knowledge Engineering (IKE03)*, Las Vegas, Nevada, USA, June, 2003, pp. 438-444
- [59] H. Zhu, "Some Issues in Role-Based Collaboration", *Proceedings of IEEE Canada Conference on Electrical and Computer Engineering (CCECE03)*, Montreal, Canada, May, 2003.
- [60] H. Zhu, M.C. Zhou, and R. Alknis, "Group Role Assignment via a Kuhn-Munkres Algorithm-based Solution", *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 42, no. 3, 2012, pp. 739-750.
- [61] H. Zhu, and M.C. Zhou, "Efficient Role Transfer Based on Kuhn–Munkres Algorithm", *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 42, no.2, 2012, pp. 491 - 496.
- [62] H. Zhu, "Group Role Assignment with Conflicting Agent Constraints", in *Proc. of ACM/IEEE Int'l Conf. on Collaborative Technologies and Systems*, Philadelphia, PL, USA, May 23-27, 2011, pp. 516-523.
- [63] H. Zhu and M.C. Zhou, "Role Transfer Problems and Algorithms", *IEEE Trans. on Systems, Man and Cybernetics, Part A*, vol. 36, no. 6, Nov. 2008, pp. 1442-1450.
- [64] H. W. Kuhn, "The Hungarian method for the assignment problem", *Naval Research Logistic Quarterly*, vol. 2, 1955, pp. 83-97 (Reprinted in vol. 52, no. 1, 2005, pp. 7 – 21).
- [65] J. Munkres, "Algorithms for the assignment and transportation problems", *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, no. 1, March 1957, pp. 32–38.
- [66] IBM, ILOG CPLEX Optimizer, avail: <http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/>, Aug. 2012.

PUBLISHED OR ACCEPTED PAPERS

[1] H. Zhu and L. Feng, “Agent Evaluation in Distributed Adaptive Systems”, *The IEEE International Conference on Systems, Man and Cybernetics (SMC’13)*, Toronto, Manchester, UK, Oct. 13-16, 2013(In Press).

[2] H. Zhu, L. Feng, and R. S. Grewal, “An Efficient Approach to Solving the Agent Training Problem for a Sustainable Group”, *The Summer Computer Simulation Conference (SummerSim’13)*, Toronto, Canada, July 7-10, 2013, pp. 209-216.

[3] H. Zhu and L. Feng, “An Efficient Approach to Group Role Assignment with Conflicting Agents”, *The IEEE International Conference on Computer-Supported Cooperative Work in Design (CSCWD)*, Whistler, Canada, June 27-29, 2013, pp. 145-152 .

APPENDICES

Appendix I: An AHP example

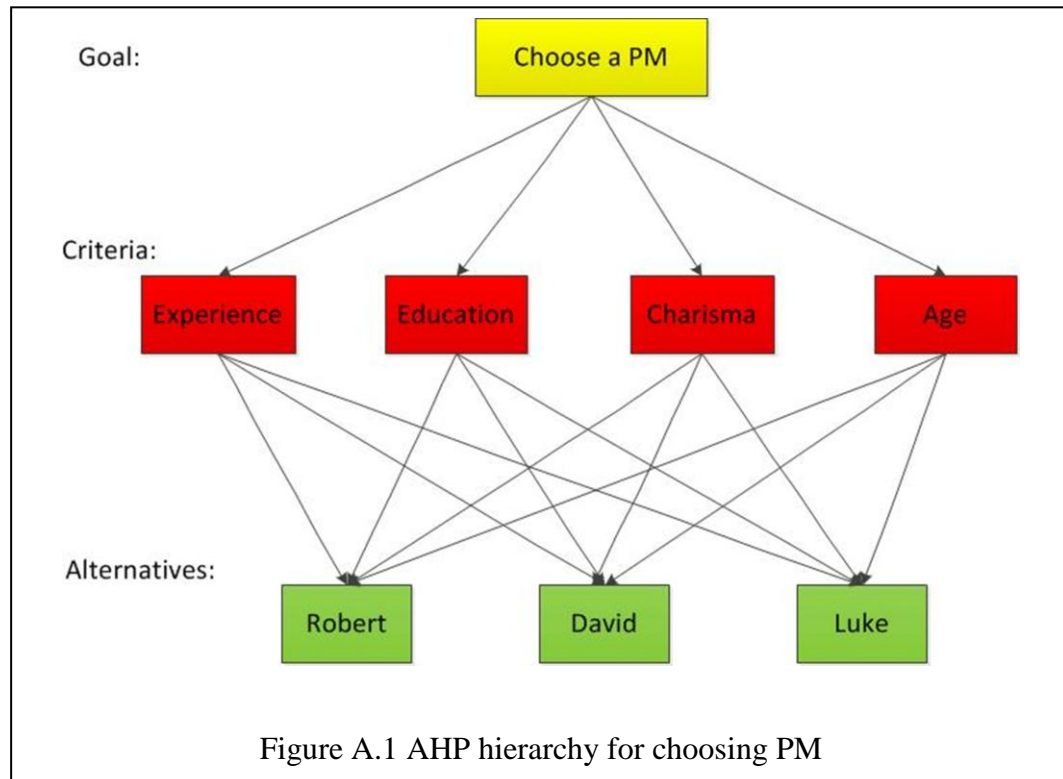
Scenario:

An IT company needs to choose a project manager (PM) from 3 candidates, Robert, David and Luke. The board considers the experience, education background, charisma and age are the most important facts of a project manager. The candidates' information is shown as below:

Table A.1: Candidates information

	Robert	David	Luke
Age	30	40	50
Experience	5 years programming, 3 years management	10 years programming, 8 years management	None programming, 20 years management, 10 years finance management
Education	BS, computer science Toronto University, MBA, McMaster University	BE, MS, software engineering, Waterloo University	BS, accountant, Ryerson University, MBA Ottawa University, Licensed CPA
Leadership	Beloved by all clients and colleagues who have worked with him	Positive example of professional work based on knowledge and experience. Was respected by everyone in company	Quiet leading style, highly respected for great performance on finance.

The AHP hierarchy for this decision making is shown below:



Pairwise comparisons

The nodes at each level need to be compared, two by two, with respect to their impact to the nodes above them. The results of these comparisons will be stored into a matrix which is calculated mathematically to derive the priorities for all the nodes on the level.

The comparisons can be made in any sequence, but in this example we will begin by comparing the criteria to the goal, and then compare the alternatives with respect to their strengths in meeting each of the Criteria.

Since there are four criteria (Experience, Education, Age, Charisma) and we need to compare each one to each of the others, we will make six pairwise comparisons with respect to each Criterion: Experience vs. Education, Experience vs. Age, Experience vs. Charisma, Education vs. Age, Education vs. Charisma and Age vs. Charisma. For each comparison, the Board will first judge which member of the pair is weaker with respect to the goal under consideration. Then they will assign a relative weight to the other criteria.

They will use the AHP Fundamental Scale in assigning the weights:

Table A.2 AHP scale weights

Intensity of importance	Definition
1	Equal importance
3	Moderate importance
5	Strong importance
7	Very strong importance
9	Extreme importance
Intensities of 2, 4, 6 and 8 can be used to indicate intermediate values.	

The compare process requires lots of debate and discussion among the decision makers. In this case, we assume that the Board agrees the following weights for the criteria:

Table A.3 Criteria weights for choosing a PM

Experience	4	Education	1
Experience	3	Charisma	1
Experience	7	Age	1
Education	1	Charisma	3
Education	3	Age	1
Age	1	Charisma	5

Now we can form a matrix based on the table above and calculate the priorities by extract the eigenvalue.

Table A.4 Priorities of criterions for choosing a PM

Criteria	Experience	Education	Charisma	Age	Priority
Experience	1	4	3	7	0.547
Education	1/4	1	1/3	3	0.127
Charisma	1/3	3	1	5	0.270
Age	1/7	1/3	1/5	1	0.056
					Sum of priorities 1.000
					Inconsistency 0.044

Similarly, we can compare the candidates two by two under each criterion. Assume that all the pair to pair comparison values are decided by the board.

Table A.5 Pairwise compare of experiences and calculated priorities

Experience	Robert	David	Luke	Priority
Robert	1	1/4	4	0.217
David	4	1	9	0.717
Luke	1/4	1/9	1	0.066
				Sum of priorities: 1.000
				Inconsistency: 0.035

Table A.6 Pairwise compare of Educations and calculated priorities

Education	Robert	David	Luke	Priority
Robert	1	3	1/5	0.188
David	1/3	1	1/7	0.081
Luke	5	7	1	0.731
				Sum of priorities: 1.000
				Inconsistency: 0.062

Table A.7 Pairwise compare of Charismas and calculated priorities

Charisma	Robert	David	Luke	Priority
Robert	1	5	9	0.743
David	1/5	1	4	0.194
Luke	1/9	1/4	1	0.063
				Sum of priorities: 1.000 Inconsistency: 0.069

Table A.8 Pairwise compare of Ages and calculated priorities

0	Robert	David	Luke	Priority
Robert	1	1/3	5	0.265
David	3	1	9	0.672
Luke	1/5	1/9	1	0.063
				Sum of priorities: 1.000 Inconsistency: 0.028

Now we have the priorities of the criteria with respect to the goal and the priorities of the alternatives with respect to the criterions. We can calculate the priorities of the alternatives with respect to the goal. It's a straightforward process of production and addition throughout the whole hierarchy.

Table A.9 Calculation of the criterions priorities for each candidate

Criterion	Priority vs. goal	Alternative	Calculation	Result
Experience	0.547	Robert	0.547×0.217	0.119
		David	0.547×0.717	0.392
		Luke	0.547×0.066	0.036
Education	0.127	Robert	0.127×0.188	0.024
		David	0.127×0.081	0.010
		Luke	0.127×0.731	0.093
Charisma	0.270	Robert	0.270×0.743	0.201
		David	0.270×0.194	0.052
		Luke	0.270×0.063	0.017
Age	0.056	Robert	0.056×0.265	0.015
		David	0.056×0.672	0.038
		Luke	0.056×0.063	0.004

For each candidate, we can calculate its total priority by adding all of its priorities under each criterion. The table below shows the result:

Table A.10 Calculation of the general priorities for each candidate

Candidate	Experience	Education	Charisma	Age	Goal
Robert	0.119	0.024	0.201	0.015	0.358
David	0.392	0.010	0.052	0.038	0.492
Luke	0.036	0.093	0.017	0.004	0.149
Total:	0.547	0.127	0.270	0.056	1.000

Appendix II: Hungarian(K-M) Algorithm solution in JAVA

```
package munkres;
import java.text.DecimalFormat;
import java.util.*;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
public class munkres {
    private static int nrow;// = 10;
    private static int ncol; // = 10;
    private static double[][] C; // = new float[nrow][ncol];
    //public static int[][] C_Orig= new int[nrow][ncol];
    private static int[][] M; // = new int[nrow][ncol];
    private static int[][] path; // = new int[ncol+nrow+1][2];
    private static int path_count = 0;
    private static int[] RowCover ; // = new int[nrow];
    private static int[] ColCover ; // = new int[ncol];
    private static int path_row_0;
    private static int path_col_0;

    public munkres(int m, int n, double[][] Cost)
    {
        nrow = m;
        ncol = n;
        M = new int[nrow][ncol];
        for(int r=0; r<nrow; r++)for(int c=0; c<ncol; c++){M[r][c] = 0;}
        C = new double[nrow][ncol];
        C = Cost;
        path = new int[ncol+nrow+1][2];

        RowCover = new int[nrow];
        ColCover = new int[ncol];
        for(int r=0; r<nrow; r++) RowCover[r] = 0;
        for(int c=0; c<ncol; c++) ColCover[c] = 0;
    }
}
```

```

public static int[][] RunMunkres(){
    int[][] result = new int [ncol][2];
    Boolean done = false;
    int step = 1;
    while(!done)
    {
        switch(step)
        {
            case 1:
                step = step_one(step);
                break;
            case 2:
                step = step_two(step);
                break;
            case 3:
                step = step_three(step);
                if (step == 7);
                break;
            case 4:
                step = step_four(step);
                break;
            case 5:
                step = step_five(step);
                break;
            case 6:
                step = step_six(step);
                break;
            case 7:
                step = step_seven(step);
                done = true;
                break;
        }
    }
    /*Find out result
    for(int c=0; c<ncol; c++)
    {
        for(int r=0; r<nrow; r++)
        {
            if (M[r][c] == 1)
            {
                result[c][0] = r;
                result[c][1] = c;
            }
        }
    }
    return result;
    */
    return M;
} //End of RunMunkres

private static int step_seven(int step) {
    //System.out.println("*****Complete*****");
    return 0;
}

//Add the vlue found in step 4 to every element of each covered row, and subtract
//it from every element of each uncoverd column. Return to step 4 without
//alternating any stars, primes, or covered lines.
private static int step_six(int step) {
    //System.out.println("Step 6"); //Test
    double minval = find_smallest();
    /*try {
        BufferedWriter out = new BufferedWriter(new
FileWriter("GRACCdata.txt",true));
        out.write("\nTHE MINI FOUND is: " + minval + "\n");
        out.close();
    }
    catch (IOException e) {System.out.println ("Error in writing into a
file!");}*/
    //System.out.println("STEP 6: the smallest value is: " +minval );
    for(int r=0; r<nrow; r++)

```

```

        for(int c=0; c<ncol; c++)
        {
            if (RowCover[r] == 1)
                C[r][c] += minval;
            if (ColCover[c] == 0)
                C[r][c] -= minval;
        }
        step = 4;
        return step;
    }

    private static double find_smallest() {
        double minval = Double.MAX_VALUE;
        for(int r=0; r<nrow; r++)
            for(int c=0; c<ncol; c++)
                if(RowCover[r] == 0 && ColCover[c] == 0)
                    if (minval>C[r][c]) minval = C[r][c];
        return minval;
    }

    //Construct a series of alternating primed and starred zeros as follows.
    //Let Z0 represent the uncovered primed zero found in step4. Let Z1 donate the
    //Starred zero in the column of Z0(if any). Let Z2 denote the primed zero in the
    row of Z1.
    //Continue until the series terminates at a primed zero that has no starred zero
    in its column.
    //Unstar each starred zero of the series, star each primed zero of the series,
    //erase all primes and uncover every line in the matrix. Return to step.3
    private static int step_five(int step) {
        //System.out.println("Step 5"); //Test
        boolean done = false;
        coord tempcoord = new coord();
        path_count = 1;
        path[path_count-1][0] = path_row_0;
        path[path_count-1][1] = path_col_0;

        while(!done)
        {
            find_star_in_col(path[path_count-1][1], tempcoord);
            if (tempcoord.row > -1)
            {
                path_count += 1;
                path[path_count-1][0] = tempcoord.row;
                path[path_count-1][1] = path[path_count-2][1];
            }
            else
            {
                done = true;
                if(!done)
                {
                    find_primed_in_row(path[path_count-1][0], tempcoord);
                    path_count += 1;
                    path[path_count-1][0] = path[path_count-2][0];
                    path[path_count-1][1] = tempcoord.col;
                }
            }
            augment_path();
            clear_covers();
            erase_primes();
            step = 3;
            //log(5);
            return step;
        }
    }

    private static void erase_primes() {
        for(int r=0; r<nrow; r++)
            for(int c=0; c<ncol; c++)
                if(M[r][c] == 2) M[r][c] = 0;
    }

    private static void clear_covers() {
        for(int r = 0; r<nrow; r++)

```



```

        RowCover[r] = 0;
        for(int c=0; c<ncol; c++)
            ColCover[c] = 0;
    }

    private static void augument_path() {
        for(int p=0; p<path_count; p++)
        {
            if(M[path[p][0]][path[p][1]] == 1) M[path[p][0]][path[p][1]] = 0;
            else
                M[path[p][0]][path[p][1]] = 1;
        }
    }

    private static void find_primed_in_row(int r, coord tempcoord) {
        tempcoord.row = r;
        for(int j=0; j<ncol; j++)
            if(M[r][j] == 2) tempcoord.col = j;
    }

    private static void find_star_in_col(int c, coord tempcoord) {
        tempcoord.row = -1;
        tempcoord.col = c;
        for(int i=0; i<nrow; i++)
            if(M[i][c] == 1) tempcoord.row = i;
    }

    //Find a nonecovered zero and prime it. If there is no starred zero in
    //the row containing this primed zero, go to step 5. Otherwise cover this
    //row and uncover the column containing the starred zero. Continue in this
    //manner until there are no uncovered zeros left.
    //Save the smallest uncovered value and go to step6
    private static int step_four(int step) {
        boolean done = false;

        while(!done)
        {
            coord tempcoord = new coord();
            find_a_zero(tempcoord);

            //System.out.println("Coord find OUT method FIND A ZERO:
            "+tempcoord.row+", "+tempcoord.col);
            if (tempcoord.row == -1)
            {
                done = true;
                step = 6;
            }
            else
            {
                try
                {
                    BufferedWriter out = new BufferedWriter(new
                    FileWriter("GRACCdata.txt",true));
                    out.write("STEP4: NONE COVERED ZERO COORD: " +
                    tempcoord.row + ", " + tempcoord.col + "\n");
                }
                catch (IOException e) {System.out.println ("Error in writing
                into a file!");}

                M[tempcoord.row][tempcoord.col] = 2;
                if(star_in_row(tempcoord.row))
                {
                    find_star_in_row(tempcoord); //
                    RowCover[tempcoord.row] = 1;
                    ColCover[tempcoord.col] = 0;
                }
                else
                {
                    done = true;
                    path_row_0 = tempcoord.row;
                    path_col_0 = tempcoord.col;
                }
            }
        }
    }

```

```

//System.out.println("path row col 0: "+path_row_0+",
"+path_col_0);
        step = 5;
    }
}
}
//log(4);
return step;
}

private static void find_star_in_row(coord tempcoord ) {
    tempcoord.col = -1;
    for(int c=0; c<ncol; c++)
    {
        if (M[tempcoord.row][c] == 1)
        {
            tempcoord.col = c;
            break;
        }
    }
}

private static boolean star_in_row(int row) {
    for(int c=0; c<ncol; c++)
    {
        if (M[row][c]==1) return true;
    }
    return false;
}

private static void find_a_zero(coord tempcoord) {
    int r = 0;
    int c;
    boolean done = false;
    //tempcoord.row = -1;
    //tempcoord.col = -1;
    while(!done)
    {
        c=0;
        while(true)
        {
            if(C[r][c] == 0 && RowCover[r] == 0 && ColCover[c] == 0)
            {
                /* TEST
                System.out.println("FIND A ZERO");//TEST
                try {
                    BufferedWriter out = new BufferedWriter(new
FileWriter("GRACCdata.txt",true));
                    out.write ("FIND A ZERO\n");
                    out.close();
                }
                catch (IOException e) {System.out.println ("Error in
writing into a file!");} //TEST
                */
                tempcoord.row = r;
                tempcoord.col = c;
                done = true;
            }
            c += 1;
            if(c>=ncol || done)
                break;
        }
        r += 1;
        if (r>=nrow) done = true;
    }
}

//Cover each column containing a starred zero. If K columns are covered,
//the starred zeros describe a complete set of unique assignments. In this

```

```

//case, go to DONE, otherwise go to step 4
private static int step_three(int step) {
    //System.out.println("Step 3"); //Test
    int colcount;
    for(int r=0; r<nrow; r++)
        for(int c=0; c<ncol; c++)
            if(M[r][c] == 1) ColCover[c] = 1;
    colcount = 0;
    for(int c=0; c<ncol; c++)
        if(ColCover[c] == 1) colcount += 1;
    if(colcount >= nrow || colcount >= ncol) step = 7;
    else step = 4;
    //log(3);
    return step;
}

// Find a zero(Z) in the resulting matrix. If there is no starred
// zero in its row or column, star Z. Repeat for each element in the
// matrix. Go to step3
private static int step_two(int step) {
    //System.out.println("Step 2"); //Test
    for(int r=0; r<nrow; r++)
        for(int c=0; c<ncol; c++)
            {
                if (C[r][c]==0 && RowCover[r]==0 && ColCover[c]==0)
                {
                    M[r][c] = 1;
                    RowCover[r] = 1;
                    ColCover[c] = 1;
                }
            }
    for(int r=0; r<nrow; r++)
        RowCover[r] = 0;
    for(int c=0; c<ncol; c++)
        ColCover[c] = 0;
    step = 3;
    //log(2);
    return step;
}

// for each row of the cost matrix, find the smallest element and subtract
//it from every element in its row. When finished, go to step2
private static int step_one(Integer step) {
    //System.out.println("Step 1"); //Test
    double min_in_row = Double.MAX_VALUE;
    for(int r=0; r<nrow; r++)
    {
        //min_in_row = C[r][0];
        for (int c=0; c<ncol; c++)
        {
            if(C[r][c] < min_in_row) min_in_row = C[r][c];
        }
        for(int c=0; c<ncol; c++)
        {
            C[r][c] -= min_in_row;
        }
    }
    step = 2;
    //log(1);
    return step;
}

private static void randomtest(int i, long[] time) {
    //prepare test data
    Random generator = new Random();
    for(int r=0; r<nrow; r++)
        for(int c=0; c<ncol; c++)
            {
                C[r][c] = 1-generator.nextFloat();
                M[r][c] = 0;
            }
}

```

```

        }
        for(int r=0; r<nrow; r++)
            RowCover[r] = 0;
        for(int c=0; c<ncol; c++)
            ColCover[c] = 0;
        try {
            BufferedWriter out = new BufferedWriter(new
FileWriter("GRACCdata.txt",true));
            out.write ("*****\n");
            out.write ("TEST LOOP: "+i+"\n");
            out.write ("*****\n");
            out.close();
        }
        catch (IOException e) {System.out.println ("Error in writing into a file!");
        }
        //Show test tada

        //Run
        long t1 = System.currentTimeMillis();
        RunMunkres();
        long t2 = System.currentTimeMillis();
        time[i] = t2-t1;
    }
}
class coord
{
    public static int row;
    public static int col;
public coord()
{
    row = -1;
    col = -1;
}
}

```

Appendix III: Solution of GRACCA(role level) and simulation program

```

/*
This package is used to solve general role assignment with conflicts problem.
Steps:
1. New a ogj or RAWCA_ILOG with proper parameters
2. Solve the problem by calling resolve, it will return a boolean result indicate whether
the reslove is success
3. Use methord: getOptimizedResult to get the result.
*/
package conflict_role_assign;
import ilog.concert.*;
import ilog.cplex.*;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.text.DecimalFormat;
import java.util.*;
public class RAWCA_ILOG {
    private int m; //number of agents
    private int n; //number of roles
    private double[] Q; //Qualification matrix
    private int[] C; //Conflict matrix
    private int[] L; //Requirement array
    private int[][] A; //Assignment array
    DecimalFormat df = new DecimalFormat("0.00");
    double optimized_result = 0;
    boolean bILOG_result;

    public RAWCA_ILOG(int nagent, int nrole, double[][] QM, int[][] CM, int[]RA)
    {

```

```

m = nagent;
n = nrole;

Q = new double[m*n];
for(int i=0, r=0; r<m; r++) for (int c=0; c<n; c++){Q[i] = QM[r][c]; i++; }
C = new int[m*m];
for(int i=0, r=0; r<m; r++) for (int c=0; c<m; c++){C[i] = CM[r][c]; i++; }
L = new int[n];
L = RA;
A = new int[m][n];
for(int r=0; r<m; r++) for (int c=0; c<n; c++) A[r][c] = 0;
/*//LOG:
System.out.println("Qualification Matrix: ");
for (int i=0;i<m*n;i++)
{
    System.out.print(df.format(Q[i])+"    ");
    if ((i+1)%n) == 0 System.out.print("\n");
}
System.out.print("\n");
System.out.println("Requirement Array: ");
for(int i=0; i<n; i++)
{
    System.out.print(L[i]+"    ");
}
System.out.print("\n");
System.out.println("Conflict Matrix: ");
for (int i=0; i<m*m;i++)
{
    System.out.print(C[i]+"    ");
    if ((i+1)%m == 0) System.out.print("\n");
}
System.out.print("\n");
*/
}

public boolean resolve(int[][]TR)
{
    try
    {
        //Creat cplex obj
        IloCplex cplex = new IloCplex();           //initialize the cplex object

        IloIntVar[]x = cplex.intVarArray(m*n, 0, 1); //initialize the
variables array under cplex.

        cplex.addMinimize(cplex.scalProd(x, Q));    //add the optimize
objective to cplex.

        //Add Constrains:
        //Constrain type 1: unique constrains here, one person can only be
assigned on one role at one time,
        //thus there are number of 'm' constrains here need to be inserted
into the cplex obj.
        for(int i=0; i<m; i++)
        {
            IloLinearNumExpr exprUniConstrain = cplex.linearNumExpr();
            for(int j = 0; j<n; j++)
            {
                exprUniConstrain.addTerm(1, x[n*i+j]);
            }
            cplex.addLe(exprUniConstrain, 1.0);
        }

        //Constrain type 2: Add role requirement constrains,
        //the number of people assigned on each role should meet the
requirement on that role.
        //Hence, n constrains will be added.
        for (int i = 0; i<n; i++)
        {
            IloLinearNumExpr exprReqConstrain = cplex.linearNumExpr();

```

```

        for (int j = 0; j<m; j++)
        {
            exprReqConstrain.addTerm(1, x[i+j*n]);
        }
        cplex.addEq(exprReqConstrain, L[i]);
    }

    //Constrain type 3: The conflict constrains.
    //On each role which require more than one people, all the
constrains may occur on that role should be added
    //Constrain type 3: The conflict constrains.
    for (int r=0; r<n; r++) // Scan the cost matrix by column
    {
        if ( 1 < L[r] )
        {
            //Find out all the index of x on that column
            int index[] = new int[m]; //number of person
            int indexcounter = 0;
            for(int i=0; i<m*n; i++)
            {
                if(i%n==r)
                {
                    index[indexcounter]=i;
                    indexcounter++;
                }
            }
            //Add conflicts constrains on that role.
            for(int i=0; i<m*m; i++) //i size of the conflict
chart
            {
                int row = i/m;
                int col = i%m;
                if (1 == C[i])
                {
                    IloLinearNumExpr conflict =
                    conflict.addTerm(1, x[index[col]]);
                    conflict.addTerm(1, x[index[row]]);
                    cplex.addLe(conflict, 1);
                }
            }
        }
    }

    //Solve LP
    //long t1 = System.nanoTime();
    if (cplex.solve())
    {
        bILOG_result = true;
        optimized_result = cplex.getObjValue();
        //cplex.output().println("Solution status = " +
cplex.getStatus());
        //cplex.output().println("Solution value = " +
cplex.getObjValue());

        double[] val = cplex.getValues(x);
        int ncols = cplex.getNcols();
        //cplex.output().println("Num COL: " + ncols);

        cplex.output().println("Result Table: ");
        for (int j=0; j<ncols; j++)
        {
            A[j/n][j%n] = (int)val[j];
            System.out.print(A[j/n][j%n] + " ");
            TR[j/n][j%n] = A[j/n][j%n];
            //System.out.print(val[j]+ " ");
            if ((j+1)%n == 0) {System.out.print("\n");}
        }
        //TR = A;
        cplex.end();
    }

```

```

        }
        else
        {
            cplex.end();
            bILOG_result = true;
        }
        //long t2 = System.nanoTime();
        //time[0] = (t2-t1)/1000000;
    }
    catch (IloException e){System.err.println("Concert exception" + e + "
caught");}

        return(bILOG_result);
    }

    public double getOptimizedResult()
    {
        return optimized_result;
    }
}
Test program:
package TEST;

import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.text.DecimalFormat;
import java.util.Random;
import conflict_role_assign.*;

public class IRAWC_TEST {
    private static String filename;
    private static long maxtime = Long.MIN_VALUE;
    private static long mintime = Long.MAX_VALUE;
    private static long totaltime = 0;
    private static long avetime = 0;

    static //Configurations
        int m = 500;
        static int n = m/2;
        static int cycles = 100;
        static double probability = 0.25;
        static int limM = 1000; //for TEST

    public static void main(String[] args)
    {

        while(m <= limM )
        {
            maxtime = Long.MIN_VALUE;
            mintime = Long.MAX_VALUE;
            totaltime = 0;
            avetime = 0;

            long[] timetable = new long[cycles];

            filename = ("TEST M" + m + "N" + n + ".txt");
            try {
                BufferedWriter out = new BufferedWriter(new FileWriter(filename));
                out.write("ILOG TEST: \n");

                out.write("Config:\n");
                out.write("M: " + m + "\n");
                out.write("N: " + n + "\n");
                out.write("Probability: " + probability + "\n");
                out.write("Cycle: " + cycles + "\n");
                out.close();
            }

```

```

    }
    catch (IOException e) {System.out.println ("Error in writing into a
file!");}

    //Random TEST
    for(int i=0; i<cycles; i++)
        timetable[i] = randamtest(m, n, probability);

    for(int i=0; i<cycles; i++)
    {
        totaltime += timetable[i];
        if (timetable[i] > maxtime) maxtime = timetable[i];
        if (timetable[i] < mintime) mintime = timetable[i];
    }
    avetime = totaltime/cycles;

    try {
        BufferedWriter out = new BufferedWriter(new FileWriter(filename,
true));
        out.write("*****TEST RESULT*****\n");

        out.write("MAX TIME: " + maxtime + " ms\n");
        out.write("MIN TIME: " + mintime + " ms\n");
        out.write("TOTAL TIME: " + totaltime + " ms\n");
        out.write("AVE TIME: " + avetime + " ms\n");
        out.write("*****\n");
        out.write(m + " " + maxtime + " " + mintime + " " + totaltime
+" " + avetime + "\n*****\n");

        out.close();
    }
    catch (IOException e) {System.out.println ("Error in writing into a
file!");}

    //TO ANOTHER FILE
    try {
        BufferedWriter out = new BufferedWriter(new FileWriter("TEST
RECORD.txt", true));
        out.write(m + " " + maxtime + " " + mintime + " " + totaltime
+" " + avetime + "\n");
        out.close();
    }
    catch (IOException e) {System.out.println ("Error in writing into a
file!");}

    m += 10;        //TEST interval i s10
    n = m/2;
    } //END of TEST WHILE

private static long randamtest(int m, int n, double probability) {
    Random generator = new Random();
    DecimalFormat df = new DecimalFormat("0.00");

    double[][] Q = new double[m][n];
    int[][] C = new int[m][m];
    int[] L = new int[n];

    try
    {
        BufferedWriter out = new BufferedWriter(new FileWriter((filename),
true));

        out.write("Q: \n");    // Random Q
        for(int r=0; r<m; r++)
        {
            for(int c=0; c<n; c++)
            {
                Q[r][c] = generator.nextDouble();
                out.write(df.format(Q[r][c]) + " ");
            }
            out.write("\n");
        }
    }
}

```



```

// Random C
for(int i = (m-1); i>=0; i--) //Init
{
    for(int j = 0; j<=i; j++ )
    {
        int signal;
        if (i==j) signal=0;
        else if (generator.nextDouble() <= probability)
            signal = 1;
        else signal = 0;
        C[i][j] = signal;
        C[j][i] = C[i][j];
    }
}
out.write("\nC: \n"); //Log C
for(int r=0; r<m; r++)
{
    for(int c=0; c<m; c++)
    {
        out.write(C[r][c] + " ");
    }
    out.write("\n");
}

out.write("\nL: \n"); //Random L
for (int i =0; i<n; i++)
{
    L[i] = generator.nextInt(m/n)+1;
    out.write(L[i] + " ");
}
out.write("\n");

out.close();
}
catch (IOException e) {System.out.println ("Error in writing into a
file!");}

//TEST parameters:
int[][] TR = new int[m][n];
long[] time = new long[2];

//Init ILOG and resolve
RAwCA_ILOG ILOG = new RAwCA_ILOG(m, n, Q, C, L);
ILOG.resolve(TR);//ILOG.resolve(TR, time);

//LOG result:
try
{
    BufferedWriter out = new BufferedWriter(new FileWriter(filename,
true));

    out.write("\nA: \n"); // Random Q
    for(int r=0; r<m; r++)
    {
        for(int c=0; c<n; c++)
        {
            out.write(TR[r][c] + " ");
        }
        out.write("\n");
    }
    out.write("Time cost: " + time[0] + "ms\n"); // Random Q
    out.close();
}
catch (IOException e) {System.out.println ("Error in writing into a
file!");}

return time[0];
}
}

```

Appendix IV: Solution of GRACCA (group level)

```
package groupconflict;
import ilog.concert.*;
import ilog.cplex.*;
import java.util.*;

public class groupconflict {

    public static void main(String[] args) {
        System.out.println("LOG: System started.");
        long systemstarttime = System.currentTimeMillis();
        int m = 10; //m is the number of people who can be assigned in the
system.
        int n = 5; //n is the number of roles that need to be filled with
people in the system.
        double probability = 0.1; //probability of the conflict
        Random randGenerator = new Random();
        int cycles = 5; //test recurrence time
        long[] cyclestarttime = new long[cycles];
        long[] cycleinserttime = new long[cycles];
        long[] cycleendtime = new long[cycles];
        long[] resolvetime = new long[cycles];
        long[] preparetime = new long[cycles];
        long totaltime=0;
        long mintime=(long) 1e308;
        long maxtime=0;
        long avetime=0;

        System.out.println("LOG: Start the outer test for loop");
        for(int rec=0; rec<cycles; rec++)
        {
            cyclestarttime[rec]=System.currentTimeMillis();
            try {
                IloCplex cplex = new IloCplex(); //initialize the cplex object

                IloIntVar[]x = cplex.intVarArray(m*n, 0, 1); //initialize the
variables array under cplex.

                //random initialize the objective coefficient array
                double[] objvals = new double[m*n];
                for (int i=0;i<m*n;i++)
                {
                    //objvals[i] = Math.random();
                    objvals[i] = randGenerator.nextDouble();
                    //System.out.print(objvals[i]+" ");
                    //if (i%(n-1) == 0) System.out.print("\n");
                }
                System.out.print("\n");

                //add the optimize objective to cplex
                cplex.addMaximize(cplex.scalProd(x, objvals));
                System.out.println("LOG: Objective added for loop: "+rec);

                //Add unique constraints here, one person can only be assigned on
one role at one time,
                //thus there are number of 'm' constraints here need to be inserted
into the cplex obj.
                for(int i=0; i<m; i++)
                {
                    IloLinearNumExpr exprUniComstrain = cplex.linearNumExpr();
                    for(int j = 0; j<n; j++)
                    {
                        exprUniComstrain.addTerm(1, x[n*i+j]);
                    }
                    cplex.addLe(exprUniComstrain, 1.0);
                }
            }
            catch (Exception e) {
                System.out.println("Error: " + e.getMessage());
            }
            cycleinserttime[rec]=System.currentTimeMillis();
            cycleendtime[rec]=System.currentTimeMillis();
            cycleinserttime[rec]-cyclestarttime[rec]=resolvetime[rec];
            cycleinserttime[rec]-cyclestarttime[rec]=preparetime[rec];
            totaltime+=cycleinserttime[rec]-cyclestarttime[rec];
            mintime=Math.min(mintime, cycleinserttime[rec]-cyclestarttime[rec]);
            maxtime=Math.max(maxtime, cycleinserttime[rec]-cyclestarttime[rec]);
            avetime+=cycleinserttime[rec]-cyclestarttime[rec];
        }
        System.out.println("LOG: End the outer test for loop");
        long systemendtime = System.currentTimeMillis();
        long totaltime=(systemendtime-systemstarttime)/1000;
        long mintime=(long) mintime/1000;
        long maxtime=(long) maxtime/1000;
        long avetime=(long) avetime/1000;
        System.out.println("LOG: Total time: "+totaltime+" seconds");
        System.out.println("LOG: Minimum time: "+mintime+" seconds");
        System.out.println("LOG: Maximum time: "+maxtime+" seconds");
        System.out.println("LOG: Average time: "+avetime+" seconds");
    }
}
```



```

    }

    //add group constrain
    for (int i=0; i<m;i++)
    {
        for(int j=0; j<m; j++)
        {
            if (conflictArray[i*m+j] == 1) {
                IloLinearNumExpr      exprConflict      =
cplex.linearNumExpr();
                for (int col=0; col<n; col++)
                {
                    exprConflict.addTerm(1, x[i*n+col]);
                }

                for(int col=0; col<n; col++)
                {
                    exprConflict.addTerm(1, x[j*n+col]);
                }
                cplex.addLe(exprConflict, 1);
            }
        }
    }

    /*

    //Add the conflict comstraints
    System.out.println("LOG: Start add conflict constrain in loop:
"+rec);

    for (int r=0; r<n; r++) // r is the index of the role
    {
        //System.out.println("Value of r is: "+ r);
        //find index
        int index[] = new int[m]; //number of person
        int indexcounter = 0;
        for(int i=0; i<m*n; i++) //size of the role-person chart
        {
            if(i%n==r){ //i% number of role
                index[indexcounter]=i;
                //System.out.println("In Role : "+ r +"the
index of x is : "+index[indexcounter]);
                indexcounter++;
            }
        }

        for(int i=0; i<m*m; i++) //i size of the conflict chart
        {
            int row = i/m; //row belongs (0-9)
            int col = i%m; // belongs (0-9)

            if (conflictArray[i]==1){
                IloLinearNumExpr      conflict      =
cplex.linearNumExpr();
                conflict.addTerm(1, x[index[col]]);
                conflict.addTerm(1, x[index[row]]);
                cplex.addLe(conflict, 1);
            }
        }
        System.out.println("LOG: End of add conflict constrain in loop:
"+rec);
    }*/

    cycleinserttime[rec]=System.currentTimeMillis();
    if (cplex.solve()) {
        cplex.output().println("Solution      status      =      "      +
cplex.getStatus());
        cplex.output().println("Solution      value      =      "      +
cplex.getObjValue());

```

```

        double[] val = cplex.getValues(x);
        int ncols = cplex.getNcols();
        cplex.output().println("Num COL: " + ncols);

        cplex.output().println("Result Table: ");
        for (int j=0; j<ncols; j++)
        {
            cplex.output().print(val[j]+" ");
            if ((j+1)%n) == 0
                System.out.print("\n");
        }

        /*check the correctness of the solution
        boolean result = true;
        for(int i=0; i<n; i++)
        {

            double index[] = new double[m]; //number of person
            int sub_total = 0;
            for(int j=0; j<m; j++) //size of the role-person chart
            {
                index[i] = val[i+j*n];
                sub_total+=index[i];
            }
            if (sub_total != reqArray[i]) result = false;

        }*/

        cplex.end();
        cycleendtime[rec] = System.currentTimeMillis();
        preparetime[rec] = cycleinserttime[rec]-cyclestarttime[rec];
        resolvetime[rec] = cycleendtime[rec]-cycleinserttime[rec];
        totaltime+=cycleendtime[rec]-cyclestarttime[rec];
        if (cycleendtime[rec]-cyclestarttime[rec]>maxtime)      maxtime      =
cycleendtime[rec]-cyclestarttime[rec];
        if (cycleendtime[rec]-cyclestarttime[rec]<mintime)      mintime      =
cycleendtime[rec]-cyclestarttime[rec];

    }

    catch (IloException e){
        System.err.println("Concert exception" + e + " caught");
    }

}

System.out.println("");
System.out.println("LOG time statistics:");
System.out.print("Configuration:          "+"\\n"+"cycles:"+cycles+"\\n"+"Worker
number:"+m+"\\n"+"Role number:"+n+"\\n"+"Conflict rate:"+probability+"\\n");
System.out.println("The total cal time: "+totaltime+"ms");
avetime = totaltime/cycles;
System.out.println("The average cycle time: "+avetime+"ms");
System.out.println("The max cycle time: "+maxtime+"ms");
System.out.println("The min cycle time: "+mintime+"ms");

long totalcplextime=0;
long totalpreparetime=0;
for(int i=0;i<cycles;i++)
{
    totalcplextime+=resolvetime[i];
    totalpreparetime+=preparetime[i];
}

System.out.println("The average prepare time: "+totalpreparetime/cycles+"ms");
System.out.println("The average cplex time: "+totalcplextime/cycles+"ms");

long systemendtime = System.currentTimeMillis();
System.out.println("System runing time: "+(systemendtime-systemstarttime)+"ms");
}
}

```

Appendix V Solution of Agents training problem for a sustainable group and simulation program

```

package AP_ILOG;

import ilog.concert.*;
import ilog.cplex.*;

public class AdaptiveCollaboration {
    private int iM;          //number of agents
    private int iN;          //number of roles
    private int iChoice;
    private double[][] dOriQ; //Qualification matrix
    private int[][] iOriL; //Original Requirement matrix
    //private int[][] iAdaptiveAssign; //Adaptive Assignment array

    private double[] dTransQ;
    private int[] iTransL;

    private double dOptimalResult;

    public AdaptiveCollaboration(int m, int n, int c, double[][] Q, int[][] L){
        //Initialize original data
        iM = m;
        iN = n;
        iChoice = c;
        dOriQ = new double[iM][iN];
        dOriQ = Q;
        iOriL = new int[iChoice][iN];
        iOriL = L;
        dOptimalResult = 0;

        //Initialize transform data
        dTransQ = new double[iM*iN*iChoice];
        for (int i=0; i<iM; i++)
        {
            for(int j = 0; j<iN*iChoice; j++)
            {
                dTransQ[i*iN*iChoice+j] = dOriQ[i][j%iN];
                System.out.print(dTransQ[i*iN*iChoice+j] + ", ");
            }
            System.out.print("\n");
        }

        iTransL = new int[iN*iChoice];
        int iCounter = 0;
        for(int i=0; i<iChoice; i++)
        {
            for(int j=0; j<iN; j++)
            {
                iTransL[iCounter] = iOriL[i][j];
                iCounter++;
            }
        }

        //initialize result data
        //iAdaptiveAssign = new int[iM][iN];
    }

    public int[][] resolve(int flag){
        double[] dOptimalMatrix = new double[iM*iN*iChoice];
        int[][] iAdaptiveAssign = new int[iM][iN];
        try
        {
            //initialize the cplex object
            IloCplex cplex = new IloCplex();
            //initialize the variables array under cplex.

```

```

IloIntVar[]x = cplex.intVarArray(iM*iN*iChoice, 0, 1);
//add the optimize objective to cplex.
if(0 == flag) cplex.addMinimize(cplex.scalProd(x, dTransQ));
else cplex.addMaximize(cplex.scalProd(x, dTransQ));

//Add Constrains:
//There are two types of constraints.
//First, in each group, all the roles must be fulfilled
//according to the requirement array.
//Second, each player only can be assigned once in each group

//Constraints type 1:
for(int i=0; i<iN*iChoice; i++)
{
    IloLinearNumExpr          exprRequireConstraint          =
cplex.linearNumExpr();
    for(int j = 0; j<iM; j++)
    {
        exprRequireConstraint.addTerm(1, x[i+j*iN*iChoice]);
    }
    cplex.addEq(exprRequireConstraint, iTransL[i]);
}

//Constrain type 2:
for (int i=0; i<iM; i++)
{
    for (int j=0; j<iChoice; j++)
    {
        IloLinearNumExpr exprUnique = cplex.linearNumExpr();
        for (int k=0; k<iN; k++)
        {
            exprUnique.addTerm(1, x[i*iChoice*iN+j*iN+k]);
        }
        cplex.addLe(exprUnique, 1);
    }
}

//Solve LP
if (cplex.solve())
{
    dOptimalResult = cplex.getObjValue();
    cplex.output().println("Solution value = " + dOptimalResult);

    double[] val = cplex.getValues(x);
    int ncols = cplex.getNcols();
    dOptimalMatrix = val;

    System.out.print("\nTotal Assign Matrix\n");
    for(int i=0; i<ncols;i++)
    {
        System.out.print(dOptimalMatrix[i]+" ");
        if((i+1)%iN*iChoice==0) System.out.print("\n");
    }

    cplex.end();
}
else{ cplex.end(); }
}
catch (IloException e){System.err.println("Concert exception" + e + "
caught");}

//Compressing the result matrix T' to final result matrix T
for(int count=0, i=0; i<iM; i++)
{
    for(int j=0; j<iN*iChoice; j++)
    {
        iAdaptiveAssign[i][j%iChoice] = (int)dOptimalMatrix[count];
        count++;
    }
}

```

```

        System.out.print("\n RETURN AP\n");
        for(int i=0; i<iM; i++)
        {
            for(int j=0; j<iN; j++)
            {
                System.out.print(iAdaptiveAssign[i][j]+" ");
                //count++;
            }
            System.out.print("\n");
        }

        return iAdaptiveAssign;
    }
}
Simulation Test code:
package TEST;

import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.text.DecimalFormat;
import java.util.Random;

import AP_ILOG.*;

public class AP_TEST {
    private static String filename;
    private static long maxtime = Long.MIN_VALUE;
    private static long mintime = Long.MAX_VALUE;
    private static long totaltime = 0;
    private static long avetime = 0;

    static //Configurations
        int m = 100;
        static int n = 25;
        static int cycles = 50;
        //static double probability = 0.20;
        static int limM = 20; //for TEST
        static int nL=1;

    public static void main(String[] args)
    {

        while(nL <= limM )
        {
            maxtime = Long.MIN_VALUE;
            mintime = Long.MAX_VALUE;
            totaltime = 0;
            avetime = 0;

            TEST_TIME[] timetable = new TEST_TIME[cycles];

            filename = ("TEST nL" + nL + ".txt");

            try {
                BufferedWriter out = new BufferedWriter(new FileWriter(filename));
                out.write("ILOG TEST: \n");

                out.write("Config:\n");
                out.write("M: " + m + "\n");
                out.write("N: " + n + "\n");
                //out.write("Probability: " + probability + "\n");
                out.write("Cycle: " + cycles + "\n");
                out.close();
            }
            catch (IOException e) {System.out.println ("Error in writing into a
file!");}

```



```

//Random TEST
for(int i=0; i<cycles; i++)
{
    timetable[i] = randamtest(m, n, nL);
}

for(int i=0; i<cycles; i++)
{
    totaltime += timetable[i].Tl;
    if (timetable[i].Tl > maxtime) maxtime = timetable[i].Tl;
    if (timetable[i].Tl < mintime) mintime = timetable[i].Tl;
}
avetime = totaltime/cycles;

try {
    BufferedWriter out = new BufferedWriter(new FileWriter(filename,
true));

    out.write("*****M1 RESULT*****\n");
    out.write("MAX TIME: " + maxtime + " ms\n");
    out.write("MIN TIME: " + mintime + " ms\n");
    out.write("TOTAL TIME: " + totaltime + " ms\n");
    out.write("AVE TIME: " + avetime + " ms\n");
    out.write("*****\n");
    out.write(nL + "          " + maxtime + " " + mintime + " " + totaltime
+"      " + avetime + "\n*****\n");
    out.close();
}
catch (IOException e) {System.out.println ("Error in writing into a
file!");}

//TO ANOTHER FILE
try {
    BufferedWriter out = new BufferedWriter(new FileWriter("TEST
RECORD.txt", true));
    out.write(nL + "          " + maxtime + " " + mintime + " " + totaltime
+"      " + avetime + "\n");
    out.close();
}
catch (IOException e) {System.out.println ("Error in writing into a
file!");}

//m += 10;      //TEST interval i s10
//n = m/5;
nL+=1;
} //END of TEST WHILE

private static TEST_TIME randamtest(int m, int n,int ic) {

    TEST_TIME tmpTime = new TEST_TIME();
    Random generator = new Random();
    DecimalFormat df = new DecimalFormat("0.00");

    double[][] Q = new double[m][n];
    int[][] L = new int[ic][n];

    try
    {
        BufferedWriter out = new BufferedWriter(new FileWriter((filename),
true));

        out.write("Q: \n");    // Random Q
        for(int r=0; r<m; r++)
        {
            for(int c=0; c<n; c++)
            {
                Q[r][c] = generator.nextDouble();
                out.write(df.format(Q[r][c]) + "      ");
            }
        }
    }
}

```

```

        out.write("\n");
    }

    System.out.print("L\n");
    out.write("\nL: \n");// Random L
    for(int i = 0; i<ic; i++) //Init
    {
        for(int j = 0; j<n; j++ )
        {
            L[i][j] = generator.nextInt(m/n)+1;
            out.write(L[i][j] + " ");
            System.out.print(L[i][j] + " ");
        }
        out.write("\n");
        System.out.print("\n");
    }

    out.close();
}
catch (IOException e) {System.out.println ("Error in writing into a
file!");}

//TEST NEW:
int[][] TR = new int[m][n];

//Init ILOG and resolve
AdaptiveCollaboration ILOG = new AdaptiveCollaboration(m, n, ic, Q, L);
long t1 = System.nanoTime();
TR = ILOG.resolve(1);
long t2 = System.nanoTime();
tmpTime.T1 = t2-t1;

//LOG result:
try
{
    BufferedWriter out = new BufferedWriter(new FileWriter(filename,
true));

    out.write("\nA: \n");// Random Q
    long totall = 0;

    out.write("\nTR1:\n");
    for(int r=0; r<m; r++)
    {
        for(int c=0; c<n; c++)
        {
            out.write(TR[r][c] + " ");
            totall +=(TR[r][c]*Q[r][c]);
        }
        out.write("\n");
    }
    out.write("Time cost: " + tmpTime.T1/1000 + "ns\n");// Random Q
    out.write("OPTIMAL RESULT IS: " + totall + "\n");
    out.close();
}
catch (IOException e) {System.out.println ("Error in writing into a
file!");}

return tmpTime;
}

}

class TEST_TIME {
    long T1;
    long T2;
    TEST_TIME() {T1=0;T2=0;}
}

```

Appendix VI: Solution for role-based evaluation and simulation

program

```
package Data;

import java.util.Vector;
public class Agent {

    private
        String agentName;
        int agentID;
        Vector vecSkill = new Vector(5);

    public Agent(){
        agentName= new String();
        agentID=0;
    }

    public void setAgentName(String name){agentName=name;}
    public String getAgentName(){return agentName;}

    public void setAgentID(int id){agentID=id;}
    public int getAgentID(){return agentID;}

    public void addAgentSkill(String skillName, double skillValue)
    {
        Skill tempSkill = new Skill();
        tempSkill.setName(skillName);
        tempSkill.setVal(skillValue);
        vecSkill.add(tempSkill);
    }

    public double getSkillValueByName(String skillname){
        double value = 0;
        for(int i=0;i<vecSkill.size();i++)
        {
            Skill tempSkill = (Skill) vecSkill.get(i);
            if (tempSkill.getName() == skillname) value = tempSkill.getVal();
        }
        return value;
    }
}

package Data;
import java.util.Vector;

public class Role {
    private String roleName = null;
    private int roleID = 0;
    Vector reqVector = new Vector();

    public Role(){
        roleName = null;
        roleID = 0;
    }

    public void setRoleName(String name){roleName=name;}
    public String getRoleName(){return roleName;}
    public void setRoleID(int ID){roleID=ID;}
    public int getRoleID(){return roleID;}

    public void addSkillRequirement(String skillName, double skillThreshold, double
skillWeight, boolean skillCritical)
```

```

        {
            SkillRequirement tempReq = new SkillRequirement();
            tempReq.setSkillName(skillName);
            tempReq.setSkillThreshold(skillThreshold);
            tempReq.setSkillWeight(skillWeight);
            tempReq.setSkillCritical(skillCritical);
            reqVector.add(tempReq);
        }

        public double getSkillWeightByName(String skillname){
            double value = 0;
            for(int i=0;i<reqVector.size();i++)
            {
                SkillRequirement tempSkill = (SkillRequirement) reqVector.get(i);
                if (tempSkill.getSkillName() == skillname) value =
tempSkill.getSkillWeight();
            }
            return value;
        }

        public double getSkillThresholdByName(String skillname){
            double value = 0;
            for(int i=0;i<reqVector.size();i++)
            {
                SkillRequirement tempSkill = (SkillRequirement) reqVector.get(i);
                if (tempSkill.getSkillName() == skillname) value =
tempSkill.getSkillThreshold();
            }
            return value;
        }

        public boolean getSkillCriticalByName(String skillname){
            boolean value = false;
            for(int i=0;i<reqVector.size();i++)
            {
                SkillRequirement tempSkill = (SkillRequirement) reqVector.get(i);
                if (tempSkill.getSkillName() == skillname) value =
tempSkill.getSkillCritical();
            }
            return value;
        }

        public int getNumOfRequirement(){
            return reqVector.size();
        }

        public String getSkillNameByID(int ID){

            SkillRequirement tempSkill = (SkillRequirement) reqVector.get(ID);
            return tempSkill.getSkillName();
        }
    }

    package Data;

    public class Skill {
        private
            String name;
            double val;

        public Skill(){name = null; val=0;}

        public void setName(String setName)
        {
            name = setName;
        }
        public String getName()
        {
            return name;
        }
    }

```

```

        public void setVal(double setVal)
        {
            val = setVal;
        }
        public double getVal()
        {
            return val;
        }
    }

package Data;

public class SkillRequirement {
    private String skillName = null;
    private double skillThreshold = 0;
    private boolean skillCritical = false;
    private double skillWeight = 0;

    public void setSkillName(String name){skillName = name;}
    public String getSkillName(){return skillName;}

    public void setSkillThreshold(double threshold){skillThreshold = threshold;}
    public double getSkillThreshold(){return skillThreshold;}

    public void setSkillCritical(boolean critical){skillCritical = critical;}
    public boolean getSkillCritical(){return skillCritical;}

    public void setSkillWeight(double weight){skillWeight = weight;}
    public double getSkillWeight(){return skillWeight;}
}

package RUN_TEST;
import Data.*;

import java.util.Random;
import java.lang.Math;
import java.text.*;

public class Runtest {
    private static int agentNum=50, roleNum=5, skillNum=5;
    static DecimalFormat df = new DecimalFormat("0.00");

    public Runtest(){}

    public static void main(String[] args){

        Agent Agentlist[] = new Agent[agentNum];
        Role Rolelist[] = new Role[roleNum];
        double QMatrix_1[][] = new double[agentNum][roleNum];
        double QMatrix_2[][] = new double[agentNum][roleNum];
        double QMatrix_3[][] = new double[agentNum][roleNum];
        double QMatrix_4[][] = new double[agentNum][roleNum];

        Random generator = new Random();

        //step: set values for Agents
        for(int i=0; i<agentNum; i++)
        {
            Agent tempAgent = new Agent();
            String name = new String( "Agent name"+i);
            tempAgent.setAgentName(name);
            tempAgent.setAgentID(i);
            //initial skills
            for(int j=0;j<5;j++)
            {
                String skillname = null ;
                if(j==0) skillname = "C++";
                if(j==1) skillname = "Network";
                if(j==2) skillname = "OS";
            }
        }
    }
}

```

```

        if(j==3) skillname = "DB";
        if(j==4) skillname = "Math";
        tempAgent.addAgentSkill(skillname,
(double) (generator.nextInt(8)+2));
    }
    Agentlist[i]=tempAgent;
    //System.out.println("Agent"+i+":");
    //System.out.println("Agent ID:"+Agentlist[i].getAgentID());
    //System.out.println("Agent Name:"+Agentlist[i].getAgentName());
}

//step: set values for roles
for(int i=0; i<roleNum; i++)
{
    Role tempRole = new Role();
    tempRole.setRoleID(i);

    if(i==0) tempRole.setRoleName("C++ programmer");
    if(i==1) tempRole.setRoleName("Network Admin");
    if(i==2) tempRole.setRoleName("DBA");
    if(i==3) tempRole.setRoleName("Analyst");
    if(i==4) tempRole.setRoleName("IT help");

    for(int j=0;j<5;j++)
    {
        String skillname = null ;
        double skillThrethold =0 ;
        double skillweight=0;
        boolean critical=false;
        if (i==0) //Define C++ programmer
        {
            if(j==0) {skillname = "C++"; skillThrethold = 8;
skillweight=0.5; critical=true;}
            if(j==1) {skillname = "Network"; skillThrethold = 4;
skillweight=0.1; critical=true;}
            if(j==2) {skillname = "OS"; skillThrethold = 5;
skillweight=0.2; critical=true;}
            if(j==3) {skillname = "DB"; skillThrethold = 6;
skillweight=0.1; critical=true;}
            if(j==4) {skillname = "Math"; skillThrethold = 5;
skillweight=0.1; critical=false;}
        }
        if (i==1) //Define Network Admin
        {
            if(j==0) {skillname = "C++"; skillThrethold = 3;
skillweight=0.1; critical=false;}
            if(j==1) {skillname = "Network"; skillThrethold = 8;
skillweight=0.6; critical=true;}
            if(j==2) {skillname = "OS"; skillThrethold = 6;
skillweight=0.1; critical=false;}
            if(j==3) {skillname = "DB"; skillThrethold = 5;
skillweight=0.1; critical=false;}
            if(j==4) {skillname = "Math"; skillThrethold = 5;
skillweight=0.1; critical=false;}
        }
        if (i==2) //Define DBA
        {
            if(j==0) {skillname = "C++"; skillThrethold = 3;
skillweight=0.1; critical=false;}
            if(j==1) {skillname = "Network"; skillThrethold = 5;
skillweight=0.1; critical=false;}
            if(j==2) {skillname = "OS"; skillThrethold = 7;
skillweight=0.3; critical=true;}
            if(j==3) {skillname = "DB"; skillThrethold = 7;
skillweight=0.4; critical=true;}
            if(j==4) {skillname = "Math"; skillThrethold = 4;
skillweight=0.1; critical=false;}
        }
        if (i==3) //Define Analyst

```

```

        {
            if(j==0) {skillname = "C++"; skillThrethold = 0;
skillweight=0.0; critical=false;}
            if(j==1) {skillname = "Network"; skillThrethold = 5;
skillweight=0.1; critical=false;}
            if(j==2) {skillname = "OS"; skillThrethold = 5;
skillweight=0.1; critical=false;}
            if(j==3) {skillname = "DB"; skillThrethold = 6;
skillweight=0.3; critical=true;}
            if(j==4) {skillname = "Math"; skillThrethold = 8;
skillweight=0.5; critical=true;}
        }
        if (i==4) //define IT help
        {
            if(j==0) {skillname = "C++"; skillThrethold = 4;
skillweight=0.2; critical=true;}
            if(j==1) {skillname = "Network"; skillThrethold = 4;
skillweight=0.2; critical=true;}
            if(j==2) {skillname = "OS"; skillThrethold = 4;
skillweight=0.2; critical=true;}
            if(j==3) {skillname = "DB"; skillThrethold = 4;
skillweight=0.2; critical=true;}
            if(j==4) {skillname = "Math"; skillThrethold = 4;
skillweight=0.2; critical=true;}
        }
        //Initial the temp role
        tempRole.addSkillRequirement(skillname, skillThrethold,
skillweight,critical);
    }
    //Add the role definition to the list
    Rolelist[i] = tempRole;
}

//step: print agents and roles
for(int i=0; i<agentNum; i++)
{
    //System.out.println("Agent"+i+":");
    //System.out.println("Agent ID:"+Agentlist[i].getAgentID());
    //System.out.println("Agent Name:"+Agentlist[i].getAgentName());
    System.out.print(Agentlist[i].getAgentID()+"
"+Agentlist[i].getAgentName()+" ");
    for(int j=0; j<5;j++)
    {
        String skillname1 = null ;
        if(j==0) skillname1 = "C++";
        if(j==1) skillname1 = "Network";
        if(j==2) skillname1 = "OS";
        if(j==3) skillname1 = "DB";
        if(j==4) skillname1 = "Math";
        //System.out.println(skillname1+" value: "+
Agentlist[i].getSkillValueByName(skillname1));
        System.out.print(Agentlist[i].getSkillValueByName(skillname1)+"
");
    }
    System.out.print("\n");
}

for(int i=0; i<roleNum; i++)
{
    //System.out.println("Role"+i+": ");
    //System.out.println("Role name: "+Rolelist[i].getRoleName());
    //System.out.println("Role ID: "+Rolelist[i].getRoleID());
    System.out.print(Rolelist[i].getRoleName()+" ");
    for(int j=0; j<skillNum;j++)
    {
        String skillname1 = null ;
        if(j==0) skillname1 = "C++";
        if(j==1) skillname1 = "Network";
        if(j==2) skillname1 = "OS";
        if(j==3) skillname1 = "DB";
        if(j==4) skillname1 = "Math";
    }
}

```

```

        //System.out.println("Skill name: "+ skillname1);
        System.out.print(Rolelist[i].getSkillWeightByName(skillname1)+"
"); //weight
        System.out.print(Rolelist[i].getSkillThresholdByName(skillname1)+"
"); //Threshold
        System.out.print(Rolelist[i].getSkillCriticalByName(skillname1)+"
"); //critical
    }
    System.out.print("\n");
}

for(int i=0;i<agentNum;i++)
{
    for(int j=0;j<roleNum;j++)
    {
        QMatrix_1[i][j]
Double.parseDouble(df.format(EvaluationMethod_WeightedSum(Agentlist[i], Rolelist[j]))); =
        QMatrix_2[i][j]
Double.parseDouble(df.format(EvaluationMethod_Distance(Agentlist[i], Rolelist[j]))); =
        QMatrix_3[i][j]
Double.parseDouble(df.format(EvaluationMethod_Area(Agentlist[i], Rolelist[j]))); =
        QMatrix_4[i][j]
Double.parseDouble(df.format(EvaluationMethod_MEW(Agentlist[i], Rolelist[j]))); =
    }
}

//step print Q1
System.out.println("Weighted Sum");
for(int i=0;i<agentNum;i++)
{
    for(int j=0;j<roleNum;j++)
    {
        System.out.print(QMatrix_1[i][j]+" ");
    }
    System.out.print("\n");
}

//step print Q2
System.out.println("Distance");
for(int i=0;i<agentNum;i++)
{
    for(int j=0;j<roleNum;j++)
    {
        System.out.print(QMatrix_2[i][j]+" ");
    }
    System.out.print("\n");
}

//step print Q3
System.out.println("Area");
for(int i=0;i<agentNum;i++)
{
    for(int j=0;j<roleNum;j++)
    {
        System.out.print(QMatrix_3[i][j]+" ");
    }
    System.out.print("\n");
}

//step print Q4
System.out.println("MEW");
for(int i=0;i<agentNum;i++)
{
    for(int j=0;j<roleNum;j++)
    {
        System.out.print(QMatrix_4[i][j]+" ");
    }
    System.out.print("\n");
}

```



```

    }
    private static double EvaluationMethod_MEW(Agent agent, Role role) {
        double result=1;
        double total = 1;
        int ReqNum = role.getNumOfRequirement();
        for(int i=0;i<ReqNum;i++)
        {
            String skillName = role.getSkillNameByID(i);
            double th = role.getSkillThresholdByName(skillName);
            double weight = role.getSkillWeightByName(skillName);
            double currentSkillValue=agent.getSkillValueByName(skillName);
            boolean b = role.getSkillCriticalByName(skillName);

            if (b==true && currentSkillValue<th )return 0;
            else if(currentSkillValue>=th)
            result=result*Math.pow(currentSkillValue,weight);
            else result=result*0;
            total=total*Math.pow(10,weight);
        }

        return result/total;
    }

    private static double EvaluationMethod_Area(Agent agent, Role role) {
        int ReqNum = role.getNumOfRequirement();
        double angle = 360/ReqNum;
        double currentAngle = 0;
        double vertex[][] = new double[ReqNum][2];
        double point[][] = new double [ReqNum][2];

        //initialize all points
        for(int i=0;i<ReqNum;i++)
        {
            String skillName = role.getSkillNameByID(i);
            double th = role.getSkillThresholdByName(skillName);
            double weight = role.getSkillWeightByName(skillName);
            boolean b = role.getSkillCriticalByName(skillName);
            double currentLength = agent.getSkillValueByName(skillName);
            vertex[i][0] = Math.cos(currentAngle)*10*weight;
            vertex[i][1] = Math.sin(currentAngle)*10*weight;
            if (b==true && currentLength<th) {return 0;}
            else if (currentLength>th)
            {
                point[i][0] = Math.cos(currentAngle)*currentLength*weight;
                point[i][1] = Math.sin(currentAngle)*currentLength*weight;
            }
            else {
                point[i][0] = 0;
                point[i][1] = 0;
            }
            currentAngle+=angle;
        }

        double areaTotal=0;
        double areaSub=0;

        int counter = 0;
        while(counter<ReqNum)
        {
            double a=0, b=0, c=0;
            double x1=0, y1=0;
            double x2=vertex[counter][0];
            double y2=vertex[counter][1];
            double x3, y3;
            if (counter==ReqNum-1) {x3=vertex[0][0];y3=vertex[0][1];}
            else {x3=vertex[counter+1][0];y3=vertex[counter+1][1];}
            a = DistanceOfAB(x1,y1, x2,y2);
            b = DistanceOfAB(x1,y1, x3,y3);
            c = DistanceOfAB(x3,y3, x2,y2);
            areaTotal += TriangleArea(a, b, c);
        }
    }

```

```

        double a1=0, b1=0, c1=0;
        double px1=0, py1=0;
        double px2=point[counter][0];
        double py2=point[counter][1];
        double px3, py3;
        if (counter==ReqNum-1) {px3=point[0][0];py3=point[0][1];}
        else {px3=point[counter+1][0];py3=point[counter+1][1];}
        a1 = DistanceOfAB(px1,py1, px2,py2);
        b1 = DistanceOfAB(px1,py1, px3,py3);
        c1 = DistanceOfAB(px3,py3, px2,py2);
        areaSub += TriangleArea(a1, b1, c1);
        counter++;
    }
    return areaSub/areaTotal;
}

private static double EvaluationMethod_WeightedSum(Agent agent, Role role) {
    //int numReq = role.getNumOfRequirement();
    double sum = 0;
    double total = 0;
    for(int i=0;i<role.getNumOfRequirement();i++)
    {
        String skillName = role.getSkillNameByID(i);
        double th = role.getSkillThresholdByName(skillName);
        double weight = role.getSkillWeightByName(skillName);
        boolean b = role.getSkillCriticalByName(skillName);
        double currentSkillValue = agent.getSkillValueByName(skillName);
        total += 10*weight;
        if (b==true && currentSkillValue<th) return 0;
        else if (currentSkillValue>=th) {sum += weight*currentSkillValue;}
        else sum+=0;
    }
    return sum/total;
}

private static double EvaluationMethod_Distance(Agent agent, Role role) {
    //int numReq = role.getNumOfRequirement();
    double sum = 0;
    double maxDistance = 0;
    for(int i=0;i<role.getNumOfRequirement();i++)
    {
        String skillName = role.getSkillNameByID(i);
        double th = role.getSkillThresholdByName(skillName);
        double weight = role.getSkillWeightByName(skillName);
        boolean b = role.getSkillCriticalByName(skillName);
        double currentSkillValue = agent.getSkillValueByName(skillName);
        maxDistance += Math.pow(10*weight, 2);

        if (b==true && currentSkillValue<th) return 0;
        else if(currentSkillValue<th)sum += 0;
        else sum += Math.pow(weight*currentSkillValue, 2);
    }
    return Math.sqrt(sum)/Math.sqrt(maxDistance);
}

public static double TriangleArea(double a, double b, double c){
    double p = (a+b+c)/2;
    double area = Math.sqrt(p*(p-a)*(p-b)*(p-c));
    return area;
}

public static double DistanceOfAB(double AX, double AY, double BX, double BY)
{
    return Math.sqrt(Math.pow((BX-AX), 2)+Math.pow((BY-AY), 2));
}
}

```

Appendix VII: Simulation of agents information

Table A.11 The simulated agents information

Agent		Abilities				
ID	Name	C++	NETWORK	OS	DB	MATH
0	name0	4	7	9	3	4
1	name1	3	5	5	2	9
2	name2	8	8	7	8	7
3	name3	7	6	4	7	2
4	name4	4	3	8	9	7
5	name5	8	5	4	3	8
6	name6	5	9	7	3	8
7	name7	4	9	2	9	2
8	name8	4	7	5	9	8
9	name9	8	8	4	9	5
10	name10	2	8	9	2	5
11	name11	8	2	9	9	8
12	name12	5	4	6	5	5
13	name13	9	5	8	6	6
14	name14	8	5	4	9	3
15	name15	4	3	5	6	9
16	name16	4	7	6	9	6
17	name17	4	5	7	9	7
18	name18	8	8	4	3	2
19	name19	5	4	6	8	3
20	name20	8	8	7	8	7
21	name21	8	7	6	9	2
22	name22	4	7	7	9	3
23	name23	4	3	8	5	6
24	name24	8	2	9	3	4
25	name25	7	5	9	8	6
26	name26	4	7	5	6	6
27	name27	8	9	6	7	6
28	name28	8	9	3	3	2
29	name29	5	3	2	4	9
30	name30	9	2	5	5	6
31	name31	8	7	7	9	9
32	name32	6	4	8	7	2
33	name33	9	9	3	9	5
34	name34	5	2	4	3	2

35	name35	7	4	4	6	4
36	name36	5	2	7	8	8
37	name37	2	8	6	9	6
38	name38	7	2	7	6	6
39	name39	7	5	6	6	5
40	name40	4	6	2	2	4
41	name41	5	6	8	4	2
42	name42	9	6	4	5	4
43	name43	4	4	5	5	8
44	name44	4	4	5	7	8
45	name45	6	5	5	3	2
46	name46	8	9	7	8	6
47	name47	2	4	6	2	4
48	name48	3	9	4	7	4
49	name49	8	8	6	2	3

Appendix VIII: Simulation result (SAW method)

Table A.12 Evaluation results calculated by the simple additive weighting method

Agent		Roles				
ID	Name	C++ programmer	Network Admin	DBA	Analyst	IT help
0	name0	0.0	0.0	0.0	0.0	0.0
1	name1	0.0	0.0	0.0	0.0	0.0
2	name2	0.77	0.78	0.76	0.0	0.76
3	name3	0.0	0.0	0.0	0.0	0.0
4	name4	0.0	0.0	0.71	0.0	0.0
5	name5	0.0	0.0	0.0	0.0	0.0
6	name6	0.0	0.74	0.0	0.0	0.0
7	name7	0.0	0.67	0.0	0.0	0.0
8	name8	0.0	0.0	0.0	0.79	0.66
9	name9	0.0	0.7	0.0	0.0	0.68
10	name10	0.0	0.62	0.0	0.0	0.0
11	name11	0.0	0.0	0.79	0.76	0.0
12	name12	0.0	0.0	0.0	0.0	0.5
13	name13	0.78	0.0	0.0	0.0	0.68
14	name14	0.0	0.0	0.0	0.0	0.0
15	name15	0.0	0.0	0.0	0.68	0.0
16	name16	0.0	0.0	0.0	0.0	0.64
17	name17	0.0	0.0	0.73	0.0	0.64
18	name18	0.0	0.56	0.0	0.0	0.0
19	name19	0.0	0.0	0.0	0.0	0.0
20	name20	0.77	0.78	0.76	0.0	0.76
21	name21	0.68	0.0	0.0	0.0	0.0
22	name22	0.0	0.0	0.68	0.0	0.0
23	name23	0.0	0.0	0.0	0.0	0.0

24	name24	0.0	0.0	0.0	0.0	0.0
25	name25	0.0	0.0	0.77	0.0	0.7
26	name26	0.0	0.0	0.0	0.0	0.56
27	name27	0.74	0.81	0.0	0.0	0.72
28	name28	0.0	0.62	0.0	0.0	0.0
29	name29	0.0	0.0	0.0	0.0	0.0
30	name30	0.0	0.0	0.0	0.0	0.0
31	name31	0.79	0.0	0.81	0.86	0.8
32	name32	0.0	0.0	0.58	0.0	0.0
33	name33	0.0	0.77	0.0	0.0	0.0
34	name34	0.0	0.0	0.0	0.0	0.0
35	name35	0.0	0.0	0.0	0.0	0.5
36	name36	0.0	0.0	0.66	0.71	0.0
37	name37	0.0	0.69	0.0	0.0	0.0
38	name38	0.0	0.0	0.0	0.0	0.0
39	name39	0.0	0.0	0.0	0.0	0.58
40	name40	0.0	0.0	0.0	0.0	0.0
41	name41	0.0	0.0	0.0	0.0	0.0
42	name42	0.0	0.0	0.0	0.0	0.56
43	name43	0.0	0.0	0.0	0.0	0.52
44	name44	0.0	0.0	0.0	0.66	0.56
45	name45	0.0	0.0	0.0	0.0	0.0
46	name46	0.77	0.83	0.76	0.0	0.76
47	name47	0.0	0.0	0.0	0.0	0.0
48	name48	0.0	0.64	0.0	0.0	0.0
49	name49	0.0	0.62	0.0	0.0	0.0

Appendix IX: Simulation result (MEW method)

Table A.13 Evaluation result calculated by multiplicative exponent weighting method

Agent		Roles				
ID	Name	C++ programmer	Network Admin	DBA	Analyst	IT help
0	name0	0.0	0.0	0.0	0.0	0.0
1	name1	0.0	0.0	0.0	0.0	0.0
2	name2	0.77	0.78	0.76	0.0	0.76
3	name3	0.0	0.0	0.0	0.0	0.0
4	name4	0.0	0.0	0.0	0.0	0.0
5	name5	0.0	0.0	0.0	0.0	0.0
6	name6	0.0	0.0	0.0	0.0	0.0
7	name7	0.0	0.0	0.0	0.0	0.0
8	name8	0.0	0.0	0.0	0.78	0.63
9	name9	0.0	0.0	0.0	0.0	0.65
10	name10	0.0	0.0	0.0	0.0	0.0
11	name11	0.0	0.0	0.0	0.0	0.0
12	name12	0.0	0.0	0.0	0.0	0.5
13	name13	0.76	0.0	0.0	0.0	0.66
14	name14	0.0	0.0	0.0	0.0	0.0

15	name15	0.0	0.0	0.0	0.0	0.0
16	name16	0.0	0.0	0.0	0.0	0.62
17	name17	0.0	0.0	0.71	0.0	0.62
18	name18	0.0	0.0	0.0	0.0	0.0
19	name19	0.0	0.0	0.0	0.0	0.0
20	name20	0.77	0.78	0.76	0.0	0.76
21	name21	0.0	0.0	0.0	0.0	0.0
22	name22	0.0	0.0	0.0	0.0	0.0
23	name23	0.0	0.0	0.0	0.0	0.0
24	name24	0.0	0.0	0.0	0.0	0.0
25	name25	0.0	0.0	0.76	0.0	0.69
26	name26	0.0	0.0	0.0	0.0	0.55
27	name27	0.73	0.8	0.0	0.0	0.71
28	name28	0.0	0.0	0.0	0.0	0.0
29	name29	0.0	0.0	0.0	0.0	0.0
30	name30	0.0	0.0	0.0	0.0	0.0
31	name31	0.79	0.0	0.8	0.86	0.79
32	name32	0.0	0.0	0.0	0.0	0.0
33	name33	0.0	0.0	0.0	0.0	0.0
34	name34	0.0	0.0	0.0	0.0	0.0
35	name35	0.0	0.0	0.0	0.0	0.49
36	name36	0.0	0.0	0.0	0.0	0.0
37	name37	0.0	0.0	0.0	0.0	0.0
38	name38	0.0	0.0	0.0	0.0	0.0
39	name39	0.0	0.0	0.0	0.0	0.58
40	name40	0.0	0.0	0.0	0.0	0.0
41	name41	0.0	0.0	0.0	0.0	0.0
42	name42	0.0	0.0	0.0	0.0	0.53
43	name43	0.0	0.0	0.0	0.0	0.5
44	name44	0.0	0.0	0.0	0.0	0.54
45	name45	0.0	0.0	0.0	0.0	0.0
46	name46	0.77	0.82	0.76	0.0	0.75
47	name47	0.0	0.0	0.0	0.0	0.0
48	name48	0.0	0.0	0.0	0.0	0.0
49	name49	0.0	0.0	0.0	0.0	0.0

Appendix X: Simulation result (weighted distance method)

Table A.14 Evaluation result calculated by weighted distance method

Agent		Roles				
ID	Name	C++ programmer	Network Admin	DBA	Analyst	IT help
0	name0	0.0	0.0	0.0	0.0	0.0
1	name1	0.0	0.0	0.0	0.0	0.0
2	name2	0.79	0.8	0.77	0.0	0.0
3	name3	0.0	0.0	0.0	0.0	0.0
4	name4	0.0	0.0	0.83	0.0	0.0
5	name5	0.0	0.0	0.0	0.0	0.0
6	name6	0.0	0.87	0.0	0.0	0.0

7	name7	0.0	0.87	0.0	0.0	0.0
8	name8	0.0	0.0	0.0	0.82	0.82
9	name9	0.0	0.79	0.0	0.0	0.0
10	name10	0.0	0.78	0.0	0.0	0.0
11	name11	0.0	0.0	0.88	0.82	0.82
12	name12	0.0	0.0	0.0	0.0	0.0
13	name13	0.86	0.0	0.0	0.0	0.0
14	name14	0.0	0.0	0.0	0.0	0.0
15	name15	0.0	0.0	0.0	0.81	0.81
16	name16	0.0	0.0	0.0	0.0	0.0
17	name17	0.0	0.0	0.81	0.0	0.0
18	name18	0.0	0.77	0.0	0.0	0.0
19	name19	0.0	0.0	0.0	0.0	0.0
20	name20	0.79	0.8	0.77	0.0	0.0
21	name21	0.77	0.0	0.0	0.0	0.0
22	name22	0.0	0.0	0.8	0.0	0.0
23	name23	0.0	0.0	0.0	0.0	0.0
24	name24	0.0	0.0	0.0	0.0	0.0
25	name25	0.0	0.0	0.82	0.0	0.0
26	name26	0.0	0.0	0.0	0.0	0.0
27	name27	0.77	0.88	0.0	0.0	0.0
28	name28	0.0	0.86	0.0	0.0	0.0
29	name29	0.0	0.0	0.0	0.0	0.0
30	name30	0.0	0.0	0.0	0.0	0.0
31	name31	0.79	0.0	0.83	0.89	0.89
32	name32	0.0	0.0	0.71	0.0	0.0
33	name33	0.0	0.88	0.0	0.0	0.0
34	name34	0.0	0.0	0.0	0.0	0.0
35	name35	0.0	0.0	0.0	0.0	0.0
36	name36	0.0	0.0	0.74	0.79	0.79
37	name37	0.0	0.78	0.0	0.0	0.0
38	name38	0.0	0.0	0.0	0.0	0.0
39	name39	0.0	0.0	0.0	0.0	0.0
40	name40	0.0	0.0	0.0	0.0	0.0
41	name41	0.0	0.0	0.0	0.0	0.0
42	name42	0.0	0.0	0.0	0.0	0.0
43	name43	0.0	0.0	0.0	0.0	0.0
44	name44	0.0	0.0	0.0	0.76	0.76
45	name45	0.0	0.0	0.0	0.0	0.0
46	name46	0.79	0.88	0.77	0.0	0.0
47	name47	0.0	0.0	0.0	0.0	0.0
48	name48	0.0	0.86	0.0	0.0	0.0
49	name49	0.0	0.78	0.0	0.0	0.0

Appendix XI: Simulation result (weighted area method)

TableA.15 evaluation result calculated by weighted area method

Agent	Roles
-------	-------

ID	Name	C++ programmer	Network Admin	DBA	Analyst	IT help
0	name0	0.0	0.0	0.0	0.0	0.0
1	name1	0.0	0.0	0.0	0.0	0.0
2	name2	0.1	0.17	0.2	0.0	0.57
3	name3	0.0	0.0	0.0	0.0	0.0
4	name4	0.0	0.0	0.52	0.0	0.0
5	name5	0.0	0.0	0.0	0.0	0.0
6	name6	0.0	0.45	0.0	0.0	0.0
7	name7	0.0	0.12	0.0	0.0	0.0
8	name8	0.0	0.0	0.0	0.0	0.21
9	name9	0.0	0.0	0.0	0.0	0.33
10	name10	0.0	0.0	0.0	0.0	0.0
11	name11	0.0	0.0	0.63	0.13	0.0
12	name12	0.0	0.0	0.0	0.0	0.19
13	name13	0.45	0.0	0.0	0.0	0.48
14	name14	0.0	0.0	0.0	0.0	0.0
15	name15	0.0	0.0	0.0	0.0	0.0
16	name16	0.0	0.0	0.0	0.0	0.2
17	name17	0.0	0.0	0.15	0.0	0.22
18	name18	0.0	0.0	0.0	0.0	0.0
19	name19	0.0	0.0	0.0	0.0	0.0
20	name20	0.1	0.17	0.2	0.0	0.57
21	name21	0.07	0.0	0.0	0.0	0.0
22	name22	0.0	0.0	0.01	0.0	0.0
23	name23	0.0	0.0	0.0	0.0	0.0
24	name24	0.0	0.0	0.0	0.0	0.0
25	name25	0.0	0.0	0.51	0.0	0.46
26	name26	0.0	0.0	0.0	0.0	0.14
27	name27	0.09	0.37	0.0	0.0	0.5
28	name28	0.0	0.25	0.0	0.0	0.0
29	name29	0.0	0.0	0.0	0.0	0.0
30	name30	0.0	0.0	0.0	0.0	0.0
31	name31	0.11	0.0	0.27	0.76	0.67
32	name32	0.0	0.0	0.0	0.0	0.0
33	name33	0.0	0.28	0.0	0.0	0.0
34	name34	0.0	0.0	0.0	0.0	0.0
35	name35	0.0	0.0	0.0	0.0	0.0
36	name36	0.0	0.0	0.17	0.09	0.0
37	name37	0.0	0.03	0.0	0.0	0.0
38	name38	0.0	0.0	0.0	0.0	0.0
39	name39	0.0	0.0	0.0	0.0	0.34
40	name40	0.0	0.0	0.0	0.0	0.0
41	name41	0.0	0.0	0.0	0.0	0.0
42	name42	0.0	0.0	0.0	0.0	0.07
43	name43	0.0	0.0	0.0	0.0	0.09
44	name44	0.0	0.0	0.0	0.0	0.12
45	name45	0.0	0.0	0.0	0.0	0.0
46	name46	0.11	0.62	0.18	0.0	0.54
47	name47	0.0	0.0	0.0	0.0	0.0
48	name48	0.0	0.0	0.0	0.0	0.0
49	name49	0.0	0.0	0.0	0.0	0.0